§2 Второй отборочный этап

Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго этапа составляет 2 месяца. Задачи условно разделены на задачи по математике и информатике, но носят междисциплинарный характер и в более простой форме воссоздают инженерную задачу заключительного этапа. Для каждого из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по математике, задачи по информатике - общие для всех участников. Участники не были ограничены в выборе языка программирования для решения задач.

Объем и сложность задач этого этапа подобраны таким образом, чтобы решение всех задач одним человеком было маловероятно. Это призвано обеспечить включение командной работы и распределения обязанностей. Решение каждой задачи дает определенное количество баллов. Если команда состоит из участников 9 и 10-11 классов, то команде засчитываются только те баллы по математике, которые набраны за решения задач 10-11 класса. Баллы зачисляются в полном объеме за правильное решение задачи. В данном этапе можно получить суммарно от 0 до 95 баллов.

Все условия задач по математике доступны участникам с первого дня второго отборочного этапа. Задачи по программированию выкладывались двумя партиями: в начале второго этапа и через две недели после начала. Команды могут выполнять задачи в любом порядке. Задачи допускают неограниченное число попыток сдать решение.

Задачи по математике (9 класс)

Задача 2.1.1 (3 балла)

Условие:

Преобразование плоскости называется *движением*, если оно сохраняет расстояние между точками. Композиция преобразований f и g обозначается g ∘ f: сначала применяем преобразование f, потом g.

Центральная симметрия Z_O относительно точки O - такое преобразование, что

$$X' = Z_O(X) \Leftrightarrow \overrightarrow{OX'} = -\overrightarrow{OX}.$$

Поворот R_A^α переводит точку X в такую точку X', что $\angle XOX' = \alpha$, измеренное против часовой стрелки.

Параллельный перенос $T_{\vec{a}}$ точку X переводит в такую точку X', что $\overrightarrow{XX'} = \vec{a}$.

Осевая симметрия S_l относительно прямой l переводит точку X в такую точку X', что l является серединным перпендикуляром к отрезку XX', при этом точки прямой l остаются на месте.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (*Например 01111001*).

1. Движение является параллельным переносом тогда и только тогда, когда каждый вектор оно переводит в равный ему вектор (т.е. каждый направленный отрезок переводит в эквивалентный ему);
2. Композиция двух центральных симметрий $Z_B \circ Z_A$, есть параллельный перенос на $2 \cdot \overrightarrow{AB}$;
3. Композиция двух осевых симметрий, есть параллельный перенос или поворот;
4. Композиция двух поворотов $R_A^\alpha \circ R_B^\beta$, есть поворот вокруг некоторой точки на угол $360^\circ - \alpha - \beta$;
5. Любое движение является либо параллельным переносом, либо поворотом, либо осевой симметрией, либо центральной симметрией;
6. Существует композиция из трех осевых симметрий, оставляющая все точки на своих местах;
7. $S_l \circ T_{\vec{a}} = T_{\vec{a}} \circ S_l$;
8. Для любой точки A, любого вектора $\vec{a}$ и любого угла α существуют такие прямые n и m, что $Z_A \circ T_{\vec{a}} \circ R_A^\alpha = S_n \circ S_m$.

Ответ: 11100001

Задача 2.1.2 (3 балла)

Условие:

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (*Например 01111001*).

1. Расстояние d от точки $(x_0; y_0)$ до прямой $a \cdot x + b \cdot y + c = 0$ равно
$$d = \frac{a \cdot x_0 + b \cdot y_0 + c}{\sqrt{a^2 + b^2}};$$
2. Окружность проходит через точку (2; 1) и касается осей координат. Тогда она задается уравнением $(x - 1)^2 + (y - 1)^2 = 1$;
3. В треугольнике с вершинами в точках $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$ точка пересечения медиан имеет координаты
$$\left(\frac{2x_1 + 2x_2 + 2x_3}{3};\ \frac{2y_1 + 2y_2 + 2y_3}{3}\right);$$
4. Прямая с угловым коэффициентом k проходит через точку $(x_0; y_0)$ тогда и только тогда, когда ее уравнение принимает вид
$$y - y_0 = k \cdot (x - x_0);$$
5. Две прямые, заданные уравнениями $y = k_1 \cdot x + l_1$ и $y = k_2 \cdot x + l_2$, перпендикулярны тогда и только тогда, когда $k_1 \cdot k_2 = -1$;
6. При повороте на угол α (по часовой стрелке) с центром в начале координат точка с координатами (x; y) переходит в точку
$$(x \cdot \cos\alpha - y \cdot \sin\alpha;\ x \cdot \sin\alpha + y \cdot \cos\alpha);$$
7. Множество точек, удовлетворяющих уравнению
$$x^2 + y^2 = x + y + \frac{1}{2},$$
является окружностью;
8. Площадь треугольника с вершинами в точках (0; 0), $(x_1; y_1)$ и $(x_2; y_2)$ равна

$$\frac{|x_1 \cdot y_2 - x_2 \cdot y_1|}{2}.$$

Ответ: 00011011

Задача 2.1.3 (3 балла)

Условие:

Везде далее I обозначает единичную матрицу $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Вектора удобно рассматривать матрицами, имеющими один столбец: $\vec{a}\{x; y\} = \begin{pmatrix} x \\ y \end{pmatrix}$.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (*Например 01111001*).

1. Дана матрица $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, тогда $detA = a \cdot d - b \cdot c$;
2. Для любой невырожденной матрицы A размера 2×2 выполняется равенство $$(A + A^{-1})^2 = A^2 + (A^{-1})^2 + 2 \cdot I;$$
3. Для любых матриц A и B размера 2×2 выполняется равенство $$(A \cdot B^T)^T = A^T \cdot B;$$
4. Если матрицы $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$, то $A^T \cdot B^T = \begin{pmatrix} 8 & 20 \\ 5 & 13 \end{pmatrix}$;
5. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда направленный угол $\angle(\overrightarrow{OM};\ \overrightarrow{OM'})$ равен α.
6. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда точки M и M' симметричны относительно прямой $y = \alpha \cdot x$.
7. Для любых матриц A и B размера 2×2 выполняется равенство $$(A + B)^2 = B^2 + 2 \cdot A \cdot B + A^2;$$
8. Если матрицы $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$, то $3 \cdot A - 2 \cdot B = 5 \cdot \begin{pmatrix} -1 & 0 \\ 0 & 2 \end{pmatrix}$.

Ответ: 11000100

Задача 2.1.4 (2 балла)

Условие:

Найдите сумму коэффициентов матрицы $\frac{1}{5^{100}} \cdot A^{100}$, если матрица

$$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 5 \end{pmatrix}.$$

Решение:

Пусть матрица

$$B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

тогда матрицу A можно представить в виде

$$A = 5 \cdot I + B,$$

тогда

$$A^n = (5 \cdot I + B)^n.$$

Раскроем по биному Ньютона и приведем подобные члены (проблем с приведением подобных членов не будет, так как матрицы I и B перестановочны)

$$A^n = \sum_{k=0}^{n} \frac{n!}{k! \cdot (n-k)!} \cdot 5^k \cdot I^k \cdot B^{n-k}$$

Почти все слагаемые этой суммы равны нулевой матрице, так как $B^{n-k} = 0$ при $n - k \geq 3$. Поэтому

$$A^n = 5^n \cdot I^n + n \cdot 5^{n-1} \cdot I^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot I^{n-2} \cdot B^2 =$$

$$= 5^n \cdot I + n \cdot 5^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot B^2$$

То есть матрица A^n имеет следующий вид

$$A^n = \begin{pmatrix} 5^n & n \cdot 5^{n-1} & \frac{n(n-1)}{2} \cdot 5^{n-2} \\ 0 & 5^n & n \cdot 5^{n-1} \\ 0 & 0 & 5^n \end{pmatrix} = 5^n \cdot \begin{pmatrix} 1 & \frac{n}{5} & \frac{n(n-1)}{50} \\ 0 & 1 & \frac{n}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

Следовательно,

$$\frac{1}{5^{100}} \cdot A^{100} = \begin{pmatrix} 1 & 20 & 198 \\ 0 & 1 & 20 \\ 0 & 0 & 1 \end{pmatrix}$$

Значит ответом в этой задаче будет число 1 + 1 + 1 + 20 + 20 + 198 = 241.

Ответ: 241

Задача 2.1.5 (2 балла)

Условие:

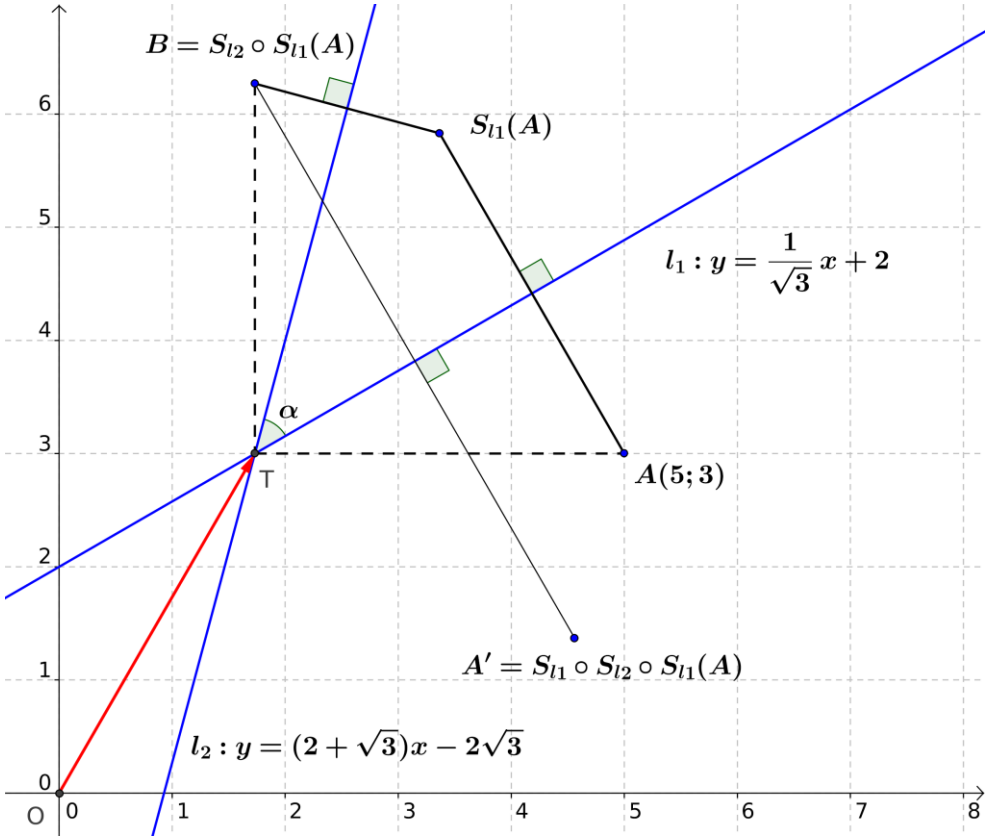
Даны две прямые $l_1: \ \sqrt{3} \cdot y = x + 2\sqrt{3}$, $l_2: \ y = (2 + \sqrt{3}) \cdot x - 2\sqrt{3}$, и точка A(5;3). Обозначим через $S_l(A)$ образ точки A при осевой симметрии относительно прямой l. Найдите координаты точки A', где

$$A' = S_{l_1} \circ S_{l_2} \circ S_{l_1}(A).$$

Значения координат округлите до тысячных. Ответ запишите в формате (-1.014; 5.000).

Решение:

Вариант 1



Пусть T - точка пересечения прямых l_1 и l_2, тогда найдем ее координаты:

$$\frac{1}{\sqrt{3}} \cdot x + 2 = (2+\sqrt{3}) \cdot x - 2\sqrt{3}$$

$$x = \frac{1+\sqrt{3}}{1+\frac{1}{\sqrt{3}}} = \frac{\sqrt{3}+3}{\sqrt{3}+1} = \sqrt{3} \qquad y = (2+\sqrt{3})\sqrt{3} - 2\sqrt{3} = 3$$

Получаем, что точка T имеет координаты $(\sqrt{3};\ 3)$. Найдем координаты точки $B = S_{l_2} \circ S_{l_1}(A)$. Так как композиция двух осевых симметрий $S_{l_2} \circ S_{l_1}$ есть не что иное, как поворот $R_T^{2\alpha}$, где α - ориентированный угол $\angle(l_1;\ l_2)$, то $B = R_T^{2\alpha}(A)$. Ввиду того, что

$$\angle(l_1;\ l_2) = \angle(Ox;\ l_2) - \angle(Ox;\ l_1),$$

то

$$\angle(l_1;\ l_2) = \arctan\frac{1}{2+\sqrt{3}} - \arctan\sqrt{3} = 75^\circ - 30^\circ = 45^\circ$$

Следовательно, $B = R_T^{90^\circ}(A)$. В данном случае координаты точки B находятся очень просто (так как точки A и T имеют одинаковую координату y). В итоге получаем $B_x = T_x = \sqrt{3}$ $B_y = T_y + |TA| = 3 + (5 - \sqrt{3}) = 8 - \sqrt{3}$.

Теперь найдем координаты точки A':

$$A' = S_{l_1} \circ S_{l_2} \circ S_{l_1}(A) = S_{l_1}(B) = T_{\overrightarrow{OT}} \circ S_{l_1} \circ T_{-\overrightarrow{OT}}(B).$$

Параллельный перенос мы делаем для того, чтобы осевая симметрия совершалась относительно прямой проходящей через начало координат. Такая осевая симметрия эквивалентна домножению слева на матрицу

$\begin{pmatrix}\cos 2\beta & \sin 2\beta \\ \sin 2\beta & -\cos 2\beta\end{pmatrix}$, где β - направленный угол между осью абсцисс и осью симметрии. Угол ..., поэтому

$$A' = T_{\overrightarrow{OT}} \circ S_{l_1} \circ T_{-\overrightarrow{OT}} \left(\begin{pmatrix} \sqrt{3} \\ 8-\sqrt{3} \end{pmatrix} \right) = T_{\overrightarrow{OT}} \circ S_{l_1} \left(\begin{pmatrix} 0 \\ 5-\sqrt{3} \end{pmatrix} \right) =$$

$$= T_{\overrightarrow{OT}} \left(\begin{pmatrix} \cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 5-\sqrt{3} \end{pmatrix} \right) = T_{\overrightarrow{OT}} \left(\begin{pmatrix} \frac{5\sqrt{3}-3}{2} \\ \frac{\sqrt{3}-5}{2} \end{pmatrix} \right) = \begin{pmatrix} \frac{7\sqrt{3}-3}{2} \\ \frac{\sqrt{3}+1}{2} \end{pmatrix}$$

Вариант 2

Как и в предыдущем решении найдем координаты точки T пересечения прямых l_1 и l_2. Точку A' можно найти следующим образом

$$A' = T_{\overrightarrow{OT}} \circ S_{l_1} \circ S_{l_2} \circ S_{l_1} \circ T_{-\overrightarrow{OT}}(A).$$

Мы знаем, что осевая симметрия относительно прямой, проходящей через начало координат эквивалентна домножению слева на матрицу $\begin{pmatrix}\cos 2\beta & \sin 2\beta \\ \sin 2\beta & -\cos 2\beta\end{pmatrix}$, где β -- направленный угол между осью абсцисс и осью симметрии. Матрица соответствующая преобразованию S_{l_1} равна $\begin{pmatrix}\cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ\end{pmatrix}$, а преобразованию S_{l_2}: $\begin{pmatrix}\cos 150^\circ & \sin 150^\circ \\ \sin 150^\circ & -\cos 150^\circ\end{pmatrix}$. Тогда матрица преобразования $S_{l_1} \circ S_{l_2} \circ S_{l_1}$ равна

$$\begin{pmatrix}\cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ\end{pmatrix} \cdot \begin{pmatrix}\cos 150^\circ & \sin 150^\circ \\ \sin 150^\circ & -\cos 150^\circ\end{pmatrix} \cdot \begin{pmatrix}\cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ\end{pmatrix} =$$

$$= \begin{pmatrix}\cos 60^\circ & \sin 60^\circ \\ \sin 60^\circ & -\cos 60^\circ\end{pmatrix} \cdot \begin{pmatrix}\cos 90^\circ & -\sin 90^\circ \\ \sin 90^\circ & \cos 90^\circ\end{pmatrix} = \begin{pmatrix}\cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ\end{pmatrix}$$

В итоге получаем

$$A' = T_{\overrightarrow{OT}} \left(\begin{pmatrix}\cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ\end{pmatrix} \cdot T_{-\overrightarrow{OT}} \left(\begin{pmatrix} 5 \\ 3 \end{pmatrix} \right) \right) =$$

$$= T_{\overrightarrow{OT}} \left(\begin{pmatrix}\cos 30^\circ & -\sin 30^\circ \\ -\sin 30^\circ & -\cos 30^\circ\end{pmatrix} \cdot \begin{pmatrix} 5-\sqrt{3} \\ 0 \end{pmatrix} \right) = T_{\overrightarrow{OT}} \left(\begin{pmatrix} \frac{5\sqrt{3}-3}{2} \\ \frac{\sqrt{3}-5}{2} \end{pmatrix} \right) = \begin{pmatrix} \frac{7\sqrt{3}-3}{2} \\ \frac{\sqrt{3}+1}{2} \end{pmatrix}$$

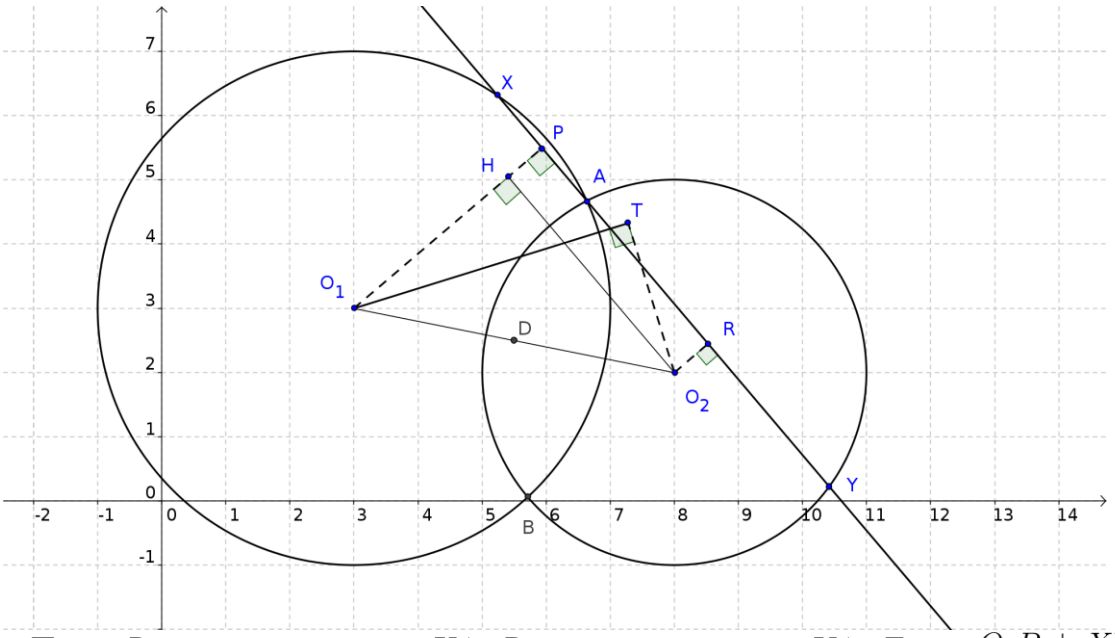
Ответ: (4,562; 1,366)

Задача 2.1.6 (2 балла)

Условие:

Даны две окружности $S_1: \ (x-3)^2 + (y-3)^2 = 16$ и $S_2: \ (x-8)^2 + (y-2)^2 = 9$, пересекающиеся в точках A и B (координаты точки A больше координат точки B). Через точку A провели две прямые l_1 и l_2. Прямая l_1 пересекает S_1 в точке X и S_2 в точке Y, а прямая l_2 пересекает S_1 в точке M и S_2 в точке N. Оказалось, что XY = MN = 8. Найдите произведение угловых коэффициентов прямых l_1 и l_2.

Решение:



Пусть P - середина хорды XA, R - середина хорды YA. Тогда $O_1P \perp XY$ и $O_2R \perp XY$. Точка H на прямой O_1P такая, что $O_2H \perp O_1P$. Тогда

$$O_2H = RP = RA + AP = \frac{XA + AY}{2} = \frac{XY}{2} = 4.$$

Аналогично находим точку T для прямой l_2 и получим $O_1T = 4$. Угловые коэффициенты прямых XY и MN равны соответственно угловым коэффициентам O_2H и O_1T. Пусть острый угол между Ox и O_1O_2 равен α и $\angle O_1O_2H = \beta$. Тогда угловой коэффициент прямой OH равен $k_1 = \tan(180 - \alpha - \beta) = -\tan(\alpha + \beta)$. А так как $\triangle O_2O_1H = \triangle O_1O_2T$, то $\angle O_2O_1T = \angle O_1O_2H = \beta$. Следовательно, угловой коэффициент прямой O_1T равен $k_2 = \tan(\beta - \alpha)$. Найдем $\tan\alpha$ и $\tan\beta$. Первый коэффициент находится из координат точек O_1 и O_2, а второй - из теоремы Пифагора и отношения катетов треугольника O_2O_1H. Получаем

$$\tan\alpha = \frac{1}{5} \text{ и } \tan\beta = \frac{\sqrt{1^2 + 5^2 - 4^2}}{4} = \frac{\sqrt{10}}{4}$$

Теперь мы можем найти ответ на задачу

$$k_1k_2 = -\tan(\alpha+\beta)\tan(\beta-\alpha) = -\frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta} \cdot \frac{\tan\beta - \tan\alpha}{1 + \tan\alpha\tan\beta} =$$
$$= \frac{\tan^2\alpha - \tan^2\beta}{1 - \tan^2\alpha\tan^2\beta} = \frac{\frac{1}{25} - \frac{10}{16}}{1 - \frac{1}{25}\cdot\frac{10}{16}} = -0.6$$

Ответ: -0,6

2.2 Первая попытка

Задачи по математике (10-11 класс)

Задача 2.2.1 (3 балла)

Условие:

Везде далее $I = (a_{i,j})$ обозначает квадратную единичную матрицу

$$a_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{else} \end{cases}.$$

Вектора удобно рассматривать матрицами, имеющими один столбец:

$$\vec{a}\{x;y\} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (Например 01111001).

1. Площадь параллелограмма образованного векторами $\begin{pmatrix} a \\ b \end{pmatrix}$ и $\begin{pmatrix} c \\ d \end{pmatrix}$ равна модулю определителя матрицы $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$;
2. Для любой невырожденной матрицы A выполняется равенство
$$(A + A^{-1})^2 = A^2 + (A^{-1})^2 + 2 \cdot I;$$
3. Для любых матриц A и B выполняется равенство
$$(A \cdot B^T)^T = A^T \cdot B;$$
4. Если матрицы A и B симметрические, то и матрица A · B симметрическая;
5. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда направленный угол $\angle(\overrightarrow{OM};\ \overrightarrow{OM'})$ равен α.
6. Для произвольной точки M(x; y) плоскости Oxy рассмотрим вектор-столбец $\overrightarrow{OM} = \begin{pmatrix} x \\ y \end{pmatrix}$. Обозначим вектор-столбец $\overrightarrow{OM'} = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \cdot \overrightarrow{OM}$. Тогда точки M и M' симметричны относительно прямой $y = \tan\alpha \cdot x$.
7. Для любых матриц A и B выполняется равенство
$$(A + B)^2 = B^2 + 2 \cdot A \cdot B + A^2;$$
8. Если матрицы A и B коммутативны, то для любых натуральных m и n матрицы A^n и B^m коммутативны.

Ответ: 11000101

Задача 2.2.2 (3 балла)

Условие:

Далее точки плоскости мы будем обозначать заглавными буквами, а соответствующие им комплексные числа маленькими. Если $a = x + iy$, то $\bar{a} = x - iy$ комплексное сопряжение a.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (*Например 01111001*).

1. Длина отрезка AB равна $(a - b) \cdot (\bar{a} - \bar{b})$;
2. Скалярное произведение векторов равно
$$\overrightarrow{AB} \cdot \overrightarrow{CD} = \frac{1}{2} \cdot (a - b) \cdot (c - d) + \frac{1}{2} \cdot (\bar{a} - \bar{b}) \cdot (\bar{c} - \bar{d});$$
3. Поворот по часовой стрелке на 90° с центром в начале координат задается уравнением $f(z) = -i \cdot z$;
4. Вектор $\overrightarrow{AB}$ параллелен вектору $\overrightarrow{CD}$ тогда и только тогда, когда $\dfrac{b-a}{d-c}$ - действительное число;
5. Вектор $\overrightarrow{AB}$ перпендикулярен вектору $\overrightarrow{CD}$ тогда и только тогда, когда
$$(\bar{b} - \bar{a}) \cdot (d - c) + (b - a) \cdot (\bar{d} - \bar{c}) = 0;$$
6. Прямая проходящая через точки A и B задается уравнением

$$(\bar{a} - \bar{b}) \cdot z + (b - a) \cdot \bar{z} + a \cdot \bar{b} + b \cdot \bar{a} = 0;$$

7. Точки A и B лежат на единичной окружности с центром в начале координат. Тогда комплексная координата основания перпендикуляра, опущенного из точки M на прямую, проходящую через точки A и B, находится по формуле
$$\frac{1}{2} \cdot (a + b + m + a \cdot b \cdot \bar{m})$$
8. Точки A, B, C и D лежат на единичной окружности с центром в начале координат, причем $AB \perp CD$. Тогда комплексная координата точки пересечения отрезков AB и CD находится по формуле
$$\frac{1}{2} \cdot (a + b + c + d).$$

Ответ: 10111001

Задача 2.2.3 (3 балла)

Условие:

A - матрица смежности неориентированного графа G = (V; E) (без петель и кратных ребер), где множество вершин V = {v_1, v_2, ..., v_n} и длины ребер равны 1. Число, стоящее на пересечении i строки и j столбца матрицы A, будем называть (i; j)-элементом.

Через tr(A) будем обозначать сумму диагональных элементов матрицы A.

Про каждое утверждение ниже выясните: верно оно или нет. В ответ запишите строку, состоящую из восьми цифр 0 или 1, где 1 соответствует верному утверждению, а 0 - неверному. (*Например 01111001*).

1. (i; j) - элемент матрицы A^3 равен количеству путей длины 4 из v_i в v_j;
2. (i; i) - элемент матрицы A^2 равен 0;
3. (i; i) - элемент матрицы A^3 равен удвоенному числу треугольников содержащих v_i;
4. Если граф G связный, то расстояние между различными вершинами v_i и v_j равно наименьшему из натуральных чисел m, для которых (i; j) - элемент матрицы A^m отличен от 0;
5. Граф G двудольный тогда и только тогда, когда матрица смежности имеет вид
$$\begin{pmatrix} 0 & \cdots & 0 & a_{1,1} & \cdots & a_{1,l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{t,1} & \cdots & a_{t,l} \\ a_{1,1} & \cdots & a_{t,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{1,l} & \cdots & a_{t,l} & 0 & \cdots & 0 \end{pmatrix} \qquad a_{i,j} \in \{0; 1\}$$
для некоторых натуральных t и l;
6. Граф G двудольный тогда и только тогда, когда для любого нечетного числа n все диагональные элементы матрицы A^n равны 0;
7. Число простых циклов длины 3 графа G равно $\frac{1}{3} \cdot \mathrm{tr}(A^3)$;
8. Сумма элементов i-ой строки равна степени вершины v_i.

Ответ: 00110101

Задача 2.2.4 (2 балла)

Условие:

Найдите сумму коэффициентов матрицы $\frac{1}{5^{100}} \cdot A^{100}$, если матрица $A = \begin{pmatrix} 5 & 3 & 1 \\ 0 & 5 & 3 \\ 0 & 0 & 5 \end{pmatrix}$.

Решение:

Пусть матрица $B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, тогда $B^2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$. Представим нашу матрицу A в виде

$$A = 5 \cdot I + 3 \cdot B + 1 \cdot B^2$$

тогда

$$A^n = (5 \cdot I + 3 \cdot B + 1 \cdot B^2)^n$$

Раскроем по обобщенному биному Ньютона и приведем подобные члены (проблем с приведением подобных членов не будет, так как матрицы I, B и B^2 перестановочны)

$$A^n = \sum_{\substack{k_j \geq 0 \\ k_1+k_2+k_3=n}} \frac{n!}{k_1! \cdot k_2! \cdot k_3!} \cdot (5I)^{k_1} \cdot (3B)^{k_2} \cdot (B^2)^{k_3} =$$

$$\sum_{\substack{k_j \geq 0 \\ k_1+k_2+k_3=n}} \frac{n!}{k_1! \cdot k_2! \cdot k_3!} \cdot 5^{k_1} \cdot 3^{k_2} \cdot I^{k_1} \cdot B^{k_2+2k_3}$$

Заметим, что $B^k = 0$ (нулевой матрице) при $k \geq 3$, поэтому почти все слагаемые бинома Ньютона равны 0

$$A^n = 5^n \cdot I^n + n \cdot 5^{n-1} \cdot 3 \cdot I^{n-1} \cdot B + \frac{n(n-1)}{2} \cdot 5^{n-2} \cdot 3^2 \cdot I^{n-2} \cdot B^2 +$$

$$+n \cdot 5^{n-1} \cdot I^{n-1} \cdot B^2 = 5^n \cdot I + 3 \cdot n \cdot 5^{n-1} \cdot B + \frac{1}{2} \cdot 5^{n-2} \cdot n \cdot (9n+1) \cdot B^2$$

То есть матрица A^n имеет следующий вид

$$A^n = \begin{pmatrix} 5^n & 3n \cdot 5^{n-1} & \frac{n(9n+1)}{2} \cdot 5^{n-2} \\ 0 & 5^n & 3n \cdot 5^{n-1} \\ 0 & 0 & 5^n \end{pmatrix} = 5^n \cdot \begin{pmatrix} 1 & \frac{3n}{5} & \frac{n(9n+1)}{50} \\ 0 & 1 & \frac{3n}{5} \\ 0 & 0 & 1 \end{pmatrix}$$

Следовательно,

$$\frac{1}{5^{100}} \cdot A^{100} = \begin{pmatrix} 1 & 60 & 1802 \\ 0 & 1 & 60 \\ 0 & 0 & 1 \end{pmatrix}$$

Значит ответом в этой задаче будет число 1 + 1 + 1 + 60 + 60 + 1802 = 1925.

Ответ: 1925

Задача 2.2.5 (2 балла)

Условие:

На комплексной плоскости даны две точки $A(2 + 4 \cdot i)$ и $B(4 + 0 \cdot i)$. Осевая симметрия $f(z) = n \cdot \overline{z} + m$ переводит точки A и B друг в друга. Найдите значение $|n|^2 + |m|^2$.

Решение:

Обозначим комплексные числа соответствующее точкам A и B за a и b соответственно. Возьмем в комплексной плоскости произвольную точку Z(z) и ее образ Z'(z') при осевой симметрии. На основании AZ = BZ' и BZ = AZ' имеем

$$(z-a)\cdot(\overline{z}-\overline{a})=(z'-b)\cdot(\overline{z'}-\overline{b})\text{ и }(z-b)\cdot(\overline{z}-\overline{b})=(z'-a)\cdot(\overline{z'}-\overline{a})$$

Вычтем из первого уравнения второе и получим

$$(a\overline{a}-b\overline{b})+z(\overline{b}-\overline{a})+\overline{z}(b-a)=(b\overline{b}-a\overline{a})+z'(\overline{a}-\overline{b})+\overline{z'}(a-b)$$

или

$$2(a\overline{a}-b\overline{b})+(z+z')(\overline{b}-\overline{a})+(\overline{z}+\overline{z'})(b-a)=0$$

Поделим обе части на $(\overline{a}-\overline{b})$

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-(z+z')-(\overline{z}+\overline{z'})\cdot\frac{a-b}{\overline{a}-\overline{b}}=0\qquad(*)$$

Так как AB||ZZ', то

$$\frac{z-z'}{\overline{z}-\overline{z'}}=\frac{a-b}{\overline{a}-\overline{b}}\qquad(**)$$

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-(z-z')-(\overline{z}+\overline{z'})\cdot\frac{z-z'}{\overline{z}-\overline{z'}}=2z'$$

или

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\frac{(z-z')(\overline{z}-\overline{z'})+(\overline{z}+\overline{z'})(z-z')}{\overline{z}-\overline{z'}}=2z'$$

Раскрываем скобки и приводим подобные члены

$$\frac{2(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\frac{2\overline{z}(z-z')}{\overline{z}-\overline{z'}}=2z'$$

Делим обе части на 2 и снова применяем (**)

$$z'=\frac{(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}-\overline{z}\cdot\frac{a-b}{\overline{a}-\overline{b}}$$

Теперь найдем n и m:

$$m=\frac{(a\overline{a}-b\overline{b})}{\overline{a}-\overline{b}}=\frac{20-16}{-2-4i}=\frac{-2+4i}{5}$$

$$n=-\frac{a-b}{\overline{a}-\overline{b}}=-\frac{-2+4i}{-2-4i}=\frac{3+4i}{5}$$

В итоге получаем

$$|n|^2+|m|^2=\frac{1}{25}\cdot(20+25)=\frac{9}{5}=1.8$$

Ответ: 1,8

Задача 2.2.6 (2 балла)

Условие:

Дана матрица $A=(a_{i,j})$ размера 100×100, причем

$$a_{i,j}=\begin{cases}1, & \text{if } i-j=\pm1 \text{ or } \pm99;\\ 0, & \text{else.}\end{cases}$$

Найдите $\mathrm{tr}(A^{20})$

Решение:

Заметим, что A является матрицей смежности неориентированного графа-цикла на 100 вершинах. Так как (i;j)-элемент матрицы $A^{20}=(b_{i,j})$ равен количеству путей длины 20 из вершины v_i в вершину v_j, то

$$\mathrm{tr}(A^{20})=\sum_{i=1}^{100}b_{i,i}=100\cdot b_{1,1}=100\cdot\{\text{ 20 } v_1v_1\}$$

Предпоследнее равенство следует из того, что $b_{i,i} = b_{j,j}$ для любых i и j (граф симметричен относительно любой вершины). Осталось найти чему равно $b_{1,1}$.

Так как 20 < 100, то не существует циклических путей совершающих полный оборот по нашему графу-циклу. Следовательно, в любом циклическом пути должно быть ровно 10 переходов по ребрам по часовой стрелке и 10 переходов по ребрам против часовой стрелки, причем любая такая последовательность переходов соответствует ровно одному пути ведущему из v_1 в v_1. Всего таких последовательностей C_{20}^{10} (из 20 переходов нужно выбрать 10 и сказать, что они будут по часовой стрелке). В итоге получаем

$$\mathrm{tr}(A^{20}) = 100 \cdot C_{20}^{10} = 100 \cdot \frac{20!}{10! \cdot 10!} = 18475600$$

Ответ: 18475600

2.3 Первая попытка
Задачи по информатике

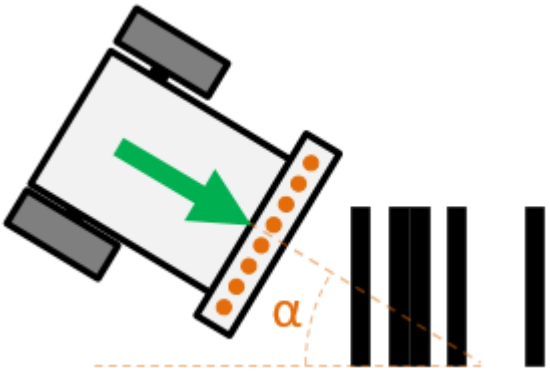
Задача 2.3.1 (10 баллов)

Условие:

Робототехническое устройство движется прямо с равномерной скоростью по плоской поверхности белого цвета. Оно оборудовано сенсором черной линии, состоящим из восьми датчиков отраженного света, расположенным на одной прямой, перпендикулярной направлению движения, на равном расстоянии друг от друга. Все датчики освещенности откалиброваны одинаково, т.е. показывают в одном и том же месте поля одинаковое значение: максимальное значение, возвращаемое датчиком на абсолютно светлой поверхности, - 100, минимальное значение, минимальное значение, возвращаемое датчиком на абсолютно черной поверхности, - 0.

В какой-то момент времени устройство начинает двигаться над участком поверхности, представляющим из себя набор из параллельных черных линий, одинаковой длины. Черные линии формируют двоичный код следующим образом:

- Первая и десятая линии являются калибровочными. Они всегда есть на поверхности поля.
- Линии со второй по девятую формируют число в двоичной системе. Если соответствующая линия есть, то она кодирует двоичную 1, если линия отсутствует, то двоичный 0.
- Вторая линия - старший бит числа, девятая линия - младший бит числа.
- Минимальное число в десятичной системе, которое может быть закодировано с помощью, данного кода - 0, максимальное - 255.



Все черные линии - одинаковой ширины.

Необходимо найти десятичное число, заданное двоичным кодом, нанесенным на участок поверхности. Угол, под которым подъезжает устройство к первой калибровочной линии, меньше 45 градусов (на рисунке обозначен α). Смещение центра сенсора черной линии относительно центра первой калибровочной линии неизвестно. Длина черных линий также неизвестна, но известно, что робототехническое устройство проезжает над кодом так, что информации, считанной со всех датчиков за время движения по поверхности с кодом, достаточно для определения кода.

Формат входных данных:

Первая строчка содержит целое число - количество замеров N $(10 \leq N \leq 100)$, выполненных сенсором черной линии.

Следующие N строчек содержат восемь целых чисел l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8 $(0 \leq l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8 \leq 100)$. Первое число l_1 - значение на крайнем левом датчике отраженного света, восьмое число l_8 - значение на крайнем правом датчике отраженного света.

Формат выходных данных:

Выведите единственное число, заданное двоичным кодом.

Пример:

stdin:

35
65 72 70 69 72 69 57 47
72 64 71 57 48 38 37 34
61 40 32 39 36 32 33 36
36 41 35 36 41 39 47 46
36 35 32 38 52 63 68 71
39 47 64 69 66 68 67 68
69 69 74 65 69 67 71 71
64 67 66 66 73 65 72 66
73 69 70 74 66 69 65 68
64 65 73 65 69 70 67 68
64 68 74 69 70 67 74 72
66 66 65 74 69 73 65 69
66 72 72 66 69 73 67 69
66 72 69 71 71 64 64 70
73 70 70 66 67 68 67 65
65 74 67 68 70 64 73 71
70 66 65 64 74 74 59 41
66 67 71 58 54 40 37 31
68 58 44 33 40 36 39 40
36 32 35 31 37 41 39 37
40 40 33 38 38 35 37 38
37 39 37 39 38 38 32 36
40 39 41 40 41 41 40 47
36 34 38 39 36 46 65 70
35 32 48 67 64 74 67 70
59 66 66 67 66 64 66 72
74 67 71 67 66 63 50 35
65 64 62 58 38 41 31 34
63 53 34 35 33 34 39 32
60 32 39 33 32 36 37 33

69 42 33 39 38 37 31 34
67 58 31 31 31 37 37 38
68 72 34 37 37 35 40 52
74 68 53 36 47 57 70 72
71 71 72 65 69 64 68 64
stdout:
13

Решение:

Решение основывается на предположении, что хотя бы один датчик из восьми прошел над всеми десятью линиями. Очевидно, что если для каждого датчика игнорировать показания, встреченные до первого и после последнего черных показаний, то датчик, в массиве которого осталось больше всего элементов – как раз тот датчик, прошедший через все линии черного цвета.

Например:

Входные данные	**Представление в виде белое/черное**
65 72 70 69 72 69 57 47	B
72 64 71 57 48 38 37 34	B B B B
61 40 32 39 36 32 33 36	B B B B B B B
36 41 35 36 41 39 47 46	B B B B B B B B
36 35 32 38 52 63 68 71	B B B B B W W W
39 47 64 69 66 68 67 68	B B W W W W W W
69 69 74 65 69 67 71 71	W W W W W W W W
64 67 66 66 73 65 72 66	W W W W W W W W
73 69 70 74 66 69 65 68	W W W W W W W W
64 65 73 65 69 70 67 68	W W W W W W W W
64 68 74 69 70 67 74 72	W W W W W W W W
66 66 65 74 69 73 65 69	W W W W W W W W
66 72 72 66 69 73 67 69	W W W W W W W W
66 72 69 71 71 64 64 70	W W W W W W W W
73 70 70 66 67 68 67 65	W W W W W W W W
65 74 67 68 70 64 73 71	W W W W W W W W
70 66 65 64 74 74 59 41	W W W W W W W B
66 67 71 58 54 40 37 31	W W W W B B B B
68 58 44 33 40 36 39 40	W W B B B B B B
36 32 35 31 37 41 39 37	B B B B B B B B
40 40 33 38 38 35 37 38	B B B B B B B B
37 39 37 39 38 38 32 36	B B B B B B B B
40 39 41 40 41 41 40 47	B B B B B B B B
36 34 38 39 36 46 65 70	B B B B B B W W
35 32 48 67 64 74 67 70	B B B W W W W W
59 66 66 67 66 64 66 72	W W W W W W W W
74 67 71 67 66 63 50 35	W W W W W B B
65 64 62 58 38 41 31 34	W W W B B B B
63 53 34 35 33 34 39 32	B B B B B B B
60 32 39 33 32 36 37 33	B B B B B B B
69 42 33 39 38 37 31 34	B B B B B B B
67 58 31 31 31 37 37 38	B B B B B B
68 72 34 37 37 35 40 52	B B B B B B

74 68 53 36 47 57 70 72 71 71 72 65 69 64 68 64	B B B

Буквой W обозначены показания выше серого (белые), B – ниже серого (черные). Серое значение следует рассчитывать как среднее арифметическое между максимальным (самым белым) и минимальным (самым черным) замерами, причем для каждого датчика отдельно.

В W/B представлении удалены лишние показания – белые замеры до встречи с первой контрольной полосой черного цвета.

Стало заметно, что максимальное полезное число замеров у датчиков под номерами 5 и 8. Можем работать с любым из них. Для примера рассмотрим датчик 8.

Количество полезных замеров – k = 33. Так как количество полос штрихкода – 10, то для успешного распознавания необходимо данные интерполировать до достижения массива длиной, кратной 10. Оптимально привести массив к длине НОК(k, 10), в нашем случае – НОК(33, 10) = 330.

При интерполяции в массив между соседними элементами записывается среднее арифметическое причем НОК(k,10)/k-1 раз, чтобы массив длиной k преобразовать в массив длиной НОК(k, 10). В нашем случае начало и конец нового массива будут выглядеть так:

0	1-9	10	11-19	20	...	320-329
47	40,5	34	35	36	...	52

После, беря среднее арифметическое для подмассивов длиной НОК(k, 10)/10 и определяя, больше ли оно серого, или нет – определяем цвет каждой полоски. Отсекаем контрольные и выводим результат, приведенный из двоичной системы счисления.

Пример программы, реализующей данный алгоритм на языке Python:

```
from fractions import gcd
# Deprecated gcd from fractions.
# gcd from math should be used
# Stepic has old version of Python,
# so has no gcd in math module.

# Arrays of white
white = [0 for k in range(8)]
# gray,
gray = [0 for k in range(8)]
# and Black values of different sensors.
black = [100 for k in range(8)]

n = int(input())
sensors = [[] for i in range(8)]

for i in range(n):
    curr_sensors = [int(x) for x in input().split()]
    for j in range(8):
        # Adding sensors data.
        sensors[j].append(curr_sensors[j])

        # Calculating white, gray and black for
        # each sensor separately.
```

```
        white[j] = max(curr_sensors[j], white[j])
        black[j] = min(curr_sensors[j], black[j])
        gray[j] = (white[j] + black[j]) / 2

# Getting rid of unwanted white values that
# preceding and succeeding important values in
# the middle. The important values start from
# the first black and ands with the last one.
max_len = 0
max_len_sens = 0

for i in range(8):
    # Going through dataset and getting
    # rid of unwanted white values.
    i_top = 0
    i_end = len(sensors[i]) - 1
    offset_top = offset_end = 0
    saw_black_top = saw_black_end = False
    while not (saw_black_top and saw_black_end):
        saw_black_top = True if sensors[i][i_top] <= gray[i] or
            i_top == len(sensors[i]) - 1 else saw_black_top
        offset_top += 1 if not saw_black_top else 0
        i_top += 1

        saw_black_end = True if sensors[i][i_end] <= gray[i] or
            i_end == 0 else saw_black_end
        offset_end += 1 if not saw_black_end else 0
        i_end -= 1

    sensors[i] = sensors[i][offset_top:len(sensors[i]) -
        offset_end]

    # Determining longest dataset, that we suppose to
    # use as dataset that came through all the stripes.
    if max_len < len(sensors[i]):
        max_len = len(sensors[i])
        max_len_sens = i

# Finding lcm of max_len and 10 helps us to
# interpolate in case max_len is not divisible by 10.
curr_len = max_len * 10 // gcd(max_len, 10)
# This array will contain the last interpolated
# (if necessary) data.
curr_data = []

# Interpolating if necessary.
if curr_len > max_len:
    for i in range(max_len):
        curr_data.append(sensors[max_len_sens][i])
        if i != max_len - 1:
            aver = (sensors[max_len_sens][i] +
                sensors[max_len_sens][i+1]) / 2
        else:
            aver = sensors[max_len_sens][i]

        for j in range(curr_len // max_len - 1):
            curr_data.append(aver)
```

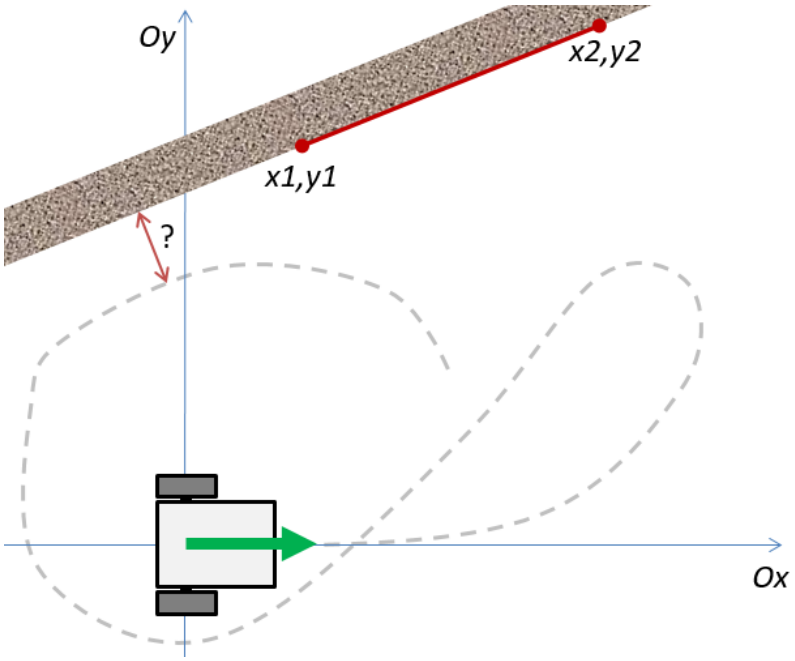
```
else:
    curr_data = sensors[max_len_sens]

ans = ""

# Making an array of "boolean" data.
# 1 - black stripe, 0 - white.
for i in range(0, curr_len, curr_len // 10):
    sum = 0
    for j in range(i, i + curr_len // 10):
        sum += curr_data[j]
    ans += str(1 if sum / (curr_len // 10) <
        gray[max_len_sens] else 0)

# Printing answer, dropping out 1st and last stripe.
# Converting binary to decimal.
print(str(int(ans[1:len(ans)-1], 2)))
```

Задача 2.3.2 (10 баллов)

Условие:

Робототехническое устройство с колеей 170 мм. и радиусом колес 28 мм, собранное по дифференциальной схеме, выезжает из точки (0,0) в направлении положительной полуоси Ox. В ходе перемещения устройство проезжает мимо стены, плоскость которой пересекает координатную плоскость по прямой, проходящей через точки (x1,y1) и (x2,y2).

Найти на каком минимальном расстоянии до стены находился во время перемещения центр робота. Ответ указать в милиметрах. Допустимая погрешность определение расстояния ± 10 мм.



Траектория перемещения робота задается, как последовательность пар чисел, определяющих показания инкрементного энкодера в градусах для левого и правого колеса через равные промежутки времени. Первое число в паре - показание энкодера левого колеса, второе число в паре - показание энкодера правого колеса.

Считать, что сцепление колес устройства с поверхностью поля - постоянное и без проскальзываний. Центр устройства совпадает с точкой, лежащей на одинаковом

расстоянии от центров пятен контактов левого и правого колес. В промежутках между измерениями показаний энкодеров скорости вращения колес постоянны, а угол поворота колеса соответствует углу поворота энкодера. Физическими силами, воздействующими на робота и его элементы, можно пренебречь.

Формат входных данных:

Первая строка входных данных содержит два целых числа x1, y1 $(-2^{31} \leq x1, y1 \leq 2^{31} - 1)$ - координаты первой точки, через которую проходит прямая.

Вторая строка содержит два целых числа x2, y2 $(-2^{31} \leq x2, y2 \leq 2^{31} - 1)$ координаты второй точки.

Третья строка содержит одно целое число n - количество показаний энкодеров $(1 \leq n \leq 2^{31} - 1)$.

В следующих n строках заданы пары чисел l и r - показания энкодера в градусах для левого и правого колеса $(-2^{31} \leq l, r \leq 2^{31} - 1)$.

Формат выходных данных:

Выведите единственное число - минимальное расстояние до стены, где во время перемещения находился центр робототехнического устройства.

Пример:

stdin:

500 -1900
-1510 -1490
41
0 0
7 7
83 90
173 150
269 208
364 266
459 325
552 383
647 441
742 500
838 559
933 617
1025 675
1122 734
1215 806
1308 867
1401 928
1490 990
1582 1051
1675 1113
1768 1174
1854 1239
1946 1302
2039 1363
2131 1425
2222 1484
2314 1546

2408 1610
2505 1675
2595 1739
2686 1804
2777 1863
2869 1924
2961 1988
3052 2050
3142 2111
3232 2176
3325 2237
3418 2298
3507 2359
3603 2433
stdout:
918

Решение:

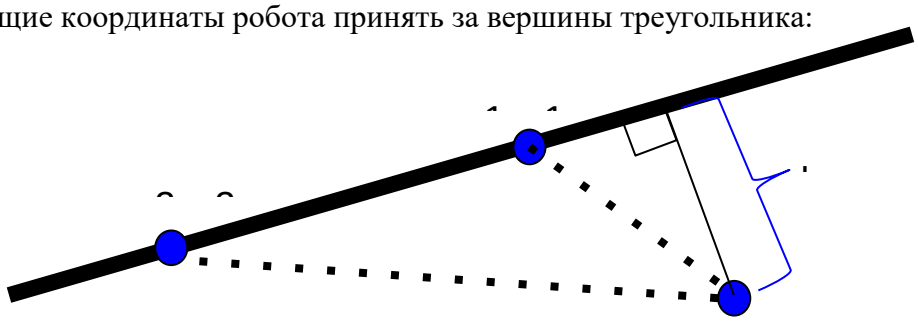
Процесс одометрии позволяет восстанавливать координаты робота относительно места его старта. Так как он стартует в точке 0,0, то эти координаты будут соответствовать абсолютным.

Зная координаты робота в любой момент времени и используя формулу расстояния между прямой и точкой на плоскости можно решить эту задачу.

Для определения расстояния между прямой и точкой на плоскости используем общую формулу площади треугольника в декартовых координатах на плоскости:

$$S = \frac{|(xB - xA)(yC - yA) - (xC - xA)(yB - yA)|}{2}$$

Если точки (x1,y1); (x2,y2) – задающие прямую, обозначающую стену , и точку (x,y) – текущие координаты робота принять за вершины треугольника:



Высота треугольника h будет обозначать кратчайшее расстояние от точки (x,y) до стены, так как падает на нее под углом 90 градусов. Так как площадь треугольника также вычисляется по формуле:

$$S = \frac{b \cdot h}{2},$$

где h – высота треугольника, а b – сторона, на которую падает высота.

Выражая искомую нами h с помощью двух уравнений и формулы расстояния между двумя точками в декартовой плоскости, получаем:

$$h = \frac{2 \cdot S}{b} = \frac{|(x2 - x1)(y - y1) - (x - x1)(y2 - y1)|}{\sqrt{(x2 - x2)^2 - (y2 - y1)^2}}$$

Для вычисления текущих координат мобильного наземного устройства, собранного по дифференциальной схеме, с воспользуемся формулами:

$$\theta_n = \theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{w}$$

$$x_n = x_{n-1} + \frac{(S_n^r - S_{n-1}^r) + (S_n^l - S_{n-1}^l)}{2} \cos(\theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{2 \cdot w})$$
$$y_n = y_{n-1} + \frac{(S_n^r - S_{n-1}^r) + (S_n^l - S_{n-1}^l)}{2} \sin(\theta_{n-1} + \frac{(S_n^r - S_{n-1}^r) - (S_n^l - S_{n-1}^l)}{2 \cdot w}),$$

где S^r и S^l - расстояния, пройденные правым и левым колесом соответственно, w - колея, а θ - угол между осью Ox и текущим направлением робота.

Пример программы, реализующей данный алгоритм на языке Java:

```
import java.io.IOException;
import java.util.Scanner;

public class Main {
    private static long w = 170; // mm - distance between wheels
    private static long r = 28; //mm  - diameter of wheels

    public static void main(String[] args) throws IOException {
        Scanner sc =new Scanner(System.in);
        long x0 = sc.nextInt();
        long y0 = sc.nextInt();
        long x1 = sc.nextInt();
        long y1 = sc.nextInt();
        int n = sc.nextInt();
        long[] encL = new long[n];
        long[] encR = new long[n];
        for (int i = 0; i < n; i++) {
            encL[i] = sc.nextInt();
            encR[i] = sc.nextInt();
        }

        // founding parameters of line( wall )
        double a = y1 - y0;
        double b = x0 - x1;
        double c = y0 * x1 - y1 * x0;
        // founding x,y and minimum distance to wall
        double x, y;
        double dist, min;

        x = 0;
        y = 0;
        min = Math.abs(c) / Math.sqrt(a * a + b * b);

        double teta = 0;
        double dSl, dSr, dTeta, dS, dX, dY;
        long dEncL, dEncR;
        for (int i = 1; i < n; i++) {
            dEncL = encL[i] - encL[i - 1];
            dEncR = encR[i] - encR[i - 1];
            dSl = (dEncL * 2 * 3.14 * r) / 360;
            dSr = (dEncR * 2 * 3.14 * r) / 360;
            dTeta = (dSr - dSl) / w;
            dS = (dSl + dSr) / 2;

            dX = dS * Math.cos(teta + dTeta / 2);
            x = x + dX;
            dY = dS * Math.sin(teta + dTeta / 2);
            y = y + dY;
            teta += dTeta;
            dist = Math.abs(a * x + b * y + c) /
```

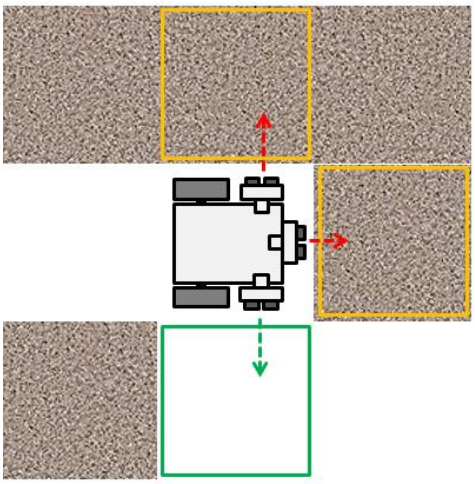
```
                    Math.sqrt(a * a + b * b);
            if (dist < min) {
                min = dist;
            }
        }

        // writing minimum distance
        System.out.println(Math.round(min));
    }
}
```

Задача 2.3.3 (15 баллов)

Условие:

После проведения ремонтных работ автоматизированный погрузчик был активирован в одном из помещений в логистическом центре. Во время ремонтных работ оперативная память погрузчика была сброшена, поэтому устройство управления данного погрузчика не имеет информации о том, в каком помещении погрузчик активирован. Тем не менее, устройство управления имеет доступ к карте всего логистического центра, которое сохранилось в постоянной памяти погрузчика.

Для навигации в помещении на устройстве установлено три датчика обнаружения препятствий: один из датчиков определяет наличие препятствия прямо перед погрузчиком, другие с левой и правой стороны, что позволяет определять препятствие слева и справа соответственно.



Если датчик видит препятствие, он возвращает 1, иначе возвращает 0.

Помещения представляют из себя квадратные секции. За одну фазу перемещения погрузчик переходит прямо из одного помещения в другое либо выполняет поворот на 90 градусов внутри текущего помещения.

Погрузчик начинает двигаться из одного помещения в другое. Необходимо определить из какого помещения началось перемещение устройства после активации.

Формат входных данных:

Первая строка входных данных содержит три целых числа n, k, m - количество помещений $(2 \leq n \leq 1000)$, количество переходов, соединяющих помещения $(1 \leq k \leq n \cdot \frac{(n-1)}{2})$ и количество перемещений, выполненных погрузчиком $(2 \leq m \leq 10000)$.

Далее идет k строк, содержащих три целых числа u, v, w - номера помещений, которые соединены двусторонним переходом $(1 \leq u \leq v \leq n)$ и направление, по которым соединены эти два помещения: 0 - проход между первым и вторым

помещениями расположен по направлению на север, 1 - на восток, 2 - на юг, 3 - на запад ($w \in [0, 1, 2, 3]$). Никакие два помещения не соединены двумя переходами, и переход не соединяет помещение с самим с собой.

Следующие m строк описывают перемещения, выполненные роботом. Каждая строка содержит три числа l, c, r - показания датчиков слева, по центру и справа погрузчика $(0 \leq l, c, r \leq 3)$. Если все три числа равны 2, то данная строчка описывает поворот налево, если все три числа равны 3, то данная строчка описывает поворот погрузчика направо. Сразу после строк с цифрами 2 или 3 (кроме последней строки) идет строка с показаниями датчиков сразу после поворота. Всего строк предоставляется столько, что ими описано перемещения устройства по всем помещениям логистического центра.

Формат выходных данных:

Выведите единственное число - номер помещения, из которого началось перемещение робота после активации. -1, если определить номер помещения невозможно (симметричная структура логистического центра).

Пример:

stdin:

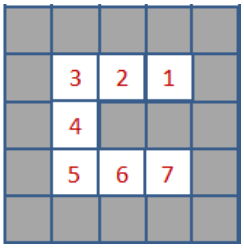
```
7 6 13
1 2 3
2 3 3
3 4 2
4 5 2
5 6 1
6 7 1
0 1 1
2 2 2
1 0 1
1 0 1
1 1 0
3 3 3
1 0 0
1 0 1
1 1 0
3 3 3
1 0 0
1 0 1
1 1 1
```

stdout:

```
7
```

Пояснение:

В примере приводится описание логистического центра, имеющего следующую структуру:



Номерами указаны помещения, по которым перемещался погрузчик.

Решение:

Для решения данной задачи можно использовать метод схожий с методом “Фильтр частиц”: мы будем итеративно оценивать все возможные положения робота с учетом текущих измерений и предыдущих перемещений:

- Шаг 1: Определить все возможные положения (размещение и направление) робота на карте с учетом первого набора показаний датчиков;
- Шаг 2: Выполнить перемещение (поворот или движение вперед) всех положений, определенных на предыдущем шаге;
- Шаг 3: Отфильтровать все положения, которые стали неактуальны из-за отсутствия возможности перемещения в соседнее помещение на предыдущем шаге.
- Шаг 4: Отфильтровать все возможные положения (расположение и направление) робота на карте с учетом текущего набора показаний датчиков;
- Повторять шаги с 2го по 4ый пока не останется только одно возможное положение. Если перемещения закончились, а возможных положений - больше чем одно, то сообщить о невозможности локализации.

Рассмотрим алгоритм действий наглядно, используя данный пример (для описания структур данных будет использоваться нотация языка Python).

Считаем v_num, e_num, path_len – кол-во комнат, переходов и длину пути робота соответственно.

Последовательно заполним массив graph[v_num + 1][4], в котором на месте i,j запишем информацию о комнате, находящейся в направлении j от комнаты i, если между ними есть переход:

Массив graph		**Номер комнаты**						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Направление перехода	**0**	-	-	-	3	4	-	-
	1	-	1	2	-	6	7	-
	2	-	-	4	5	-	-	-
	3	2	3	-	-	-	5	6

Из этого массива составим sensors_data[8][] – массив возможных показаний сенсоров робота в любом положении в любой комнате:

Показания сенсора	**Индекс в массиве**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

000	**0**								
001	**1**	3, 2	5, 1						
010	**2**	2, 0	2, 2	4, 1	4, 3	6, 0	6, 2		
011	**3**	1, 0	3, 3	5, 2	7, 0				
100	**4**	3, 1	5, 0						
101	**5**	1, 3	2, 1	2, 3	4, 0	4, 2	6, 1	6, 3	7, 3
110	**6**	1, 2	3, 0	5, 3	7, 2				
111	**7**	1, 1	7, 1						

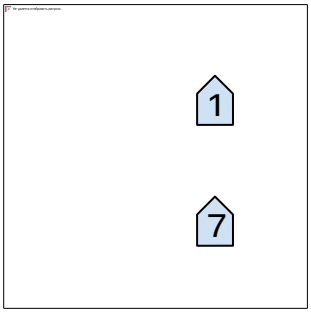
В массиве в строке i хранится информация о местоположениях (комната, направление), находясь в которых робот будет видеть датчиками определенную ситуацию, при этом i складывается из записанных слева направо показаний датчиков, переведенных из двоичной системы счисления в десятичную. Например: левый датчик видит стену (1), средний тоже (1), правый не видит (0) – полученные показания (110) при переводе из двоичной системы дают 6 – значит за всеми местоположениями, при которых датчик видит такую картину нужно обращаться в sensors_data[6].

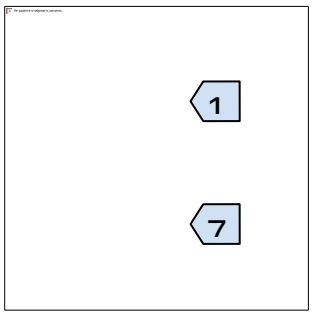
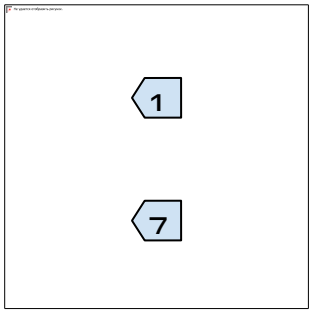
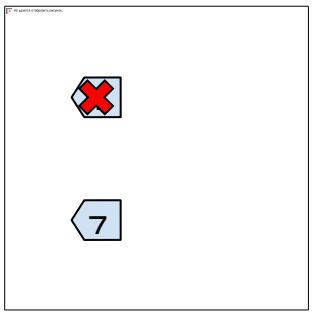
В массиве c_pos будем хранить список возможных позиций робота на текущем шаге и стартовую позицию робота. При первом запуске алгоритма заполняем его всеми возможными позициями из sensors_data[c_sens], где c_sens – первые показания сенсоров робота.

На всех остальных итерациях, пока длина c_pos не будет равна 1 выполняем следующее:

- Учитываем последние считанные данные составляем массив f_pos – массив местоположений, в которых может оказаться робот после осуществления последнего действия. В конце итераций переписываем f_pos в c_pos.

Рассмотрим пример, приведенный в условии:

№ итерации	Считанные данные	Визуализация c_pos	Комментарий
1	011		В массиве всего 2 возможных случая. В центре фигур, обозначающих возможные положения, сохраняем номер стартовой комнаты

2	222 101		Изменяем наш массив после поворота робота налево
3	101		Робот едет вперед
4	110		Ключевая точка. Показания 110 недостижимы из позиции (3, 3), а значит эту точку мы удаляем.

- Выполнять итерации дальше не имеет смысла, так как в массиве остался один элемент, задающий единственно возможное положение робота на данный момент. Так как мы хранили и стартовое положение робота, то мы легко можем дать правильный ответ – 7.

  Пример программы, реализующей данный алгоритм на языке Python:

```
# Directions: 0 - North, 1 - East,
#             2 - South, 3 - West

# Return direction after turning
# on turn_on from dir_from.
def turn(dir_from, turn_on):
    return (dir_from + turn_on + 4) % 4

# Return position after one step forward,
# if possible. Else return 0.
def step(a_v, a_dir):
    return [graph[a_v][a_dir], a_dir]
        if graph[a_v][a_dir] != 0 else 0

################################
# Program execution starts here #
################################

v_num, e_num, path_len =
    [int(x) for x in input().split()]
```

```
# graph[v][direct] = u -> vert. v has neighbour u on
# direct from it.
graph = [[0]*4 for i in range(v_num + 1)]
# List of all possible outcomes of sensors.
sensors_data = [[] for i in range(8)]

# Filling graph[v][direct] contains u - vert. num
# that is on direct from v e.g. graph[1][0] = 4 means
# there is 4th vert. on the North from the 1st vert.
for i in range(e_num):
    v, u, direct = [int(x) for x in input().split()]
    graph[v][direct] = u
    graph[u][turn(direct, 2)] = v

for v in range(1, v_num+1):       # Filling sensors_data.
    for direct in range(0, 4):
        c = "1" if graph[v][direct] == 0 else "0"
        l = "1" if graph[v][turn(direct, -1)] == 0 else "0"
        r = "1" if graph[v][turn(direct, 1)] == 0 else "0"
        sensors_data[int(l+c+r, 2)].append([v, direct])

# Current possible locations and directions: [[loc, dir],
start_loc].
# c_pos after next step of robot.
# start_loc - location, where the robot started following the
path.
c_pos = []
f_pos = []

c_step = 1

# Main algorithm starts here.
while len(c_pos) != 1 or c_step == path_len + 1:
    # Works until has 1 possible position or, if determination
failed,
    # until reaching the end of array with robots's movements.
    c_sens = "".join(input().split(" ")).replace("\r", "").\
        replace("\n", "")     # Current sensors state.

    if len(c_pos) == 0 and (c_sens != "222" and c_sens != "333"):
        c_pos = [[x, x[0]] for x in sensors_data[int(c_sens, 2)]]
    else:
        f_pos = []

        if c_sens == "222" or c_sens == "333":
            # If last action was turning, get next sensors data.
            t = -1 if c_sens == "222" else 1
            c_sens = "".join(input().split(" ")).replace("\r",
"").\
                replace("\n", "")

            for j in range(len(c_pos)):
                t_pos = [[c_pos[j][0][0], turn(c_pos[j][0][1],
t)],
                         c_pos[j][1]]
                if t_pos[0] in sensors_data[int(c_sens, 2)]:
```

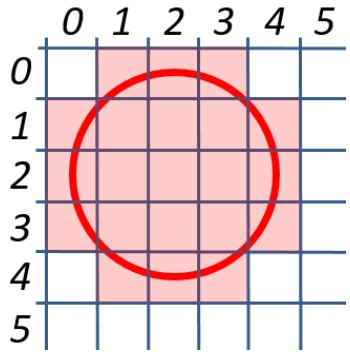
```
                    f_pos.append(t_pos)
        else:
            # Updating f_pos with positions with c_sens
visibility,
            # available from c_pos positions.
            for j in range(len(c_pos)):
                t_pos = [step(c_pos[j][0][0], c_pos[j][0][1]),
                         c_pos[j][1]]

                if t_pos[0] != 0 and t_pos[0] in \
                        sensors_data[int(c_sens, 2)]:
                    f_pos.append(t_pos)

        c_pos = f_pos    # Updating c_pos after 1 step.
    c_step += 1

if c_step <= path_len + 1:
    print(c_pos[0][1])
else:
    print(-1)
```

Задача 2.3.4 (15 баллов)

Условие:

Мобильное робототехническое устройство перемещается по полю разбитому на ячейки. Устройство из каждой клетки может перемещаться либо на одну ячейку вверх, либо на одну ячейку влево, либо на одну ячейку вниз, либо на одну ячейку вправо. Ни одна часть устройства не может выезжать за пределы поля.

На поверхности поля нанесено N окружностей про каждую окружность известна координата ячейки - центра окружности и ее радиус, также измеряемый в ячейках. Робототехническое устройство может проезжать по тем ячейкам поля, которые заняты окружностями.



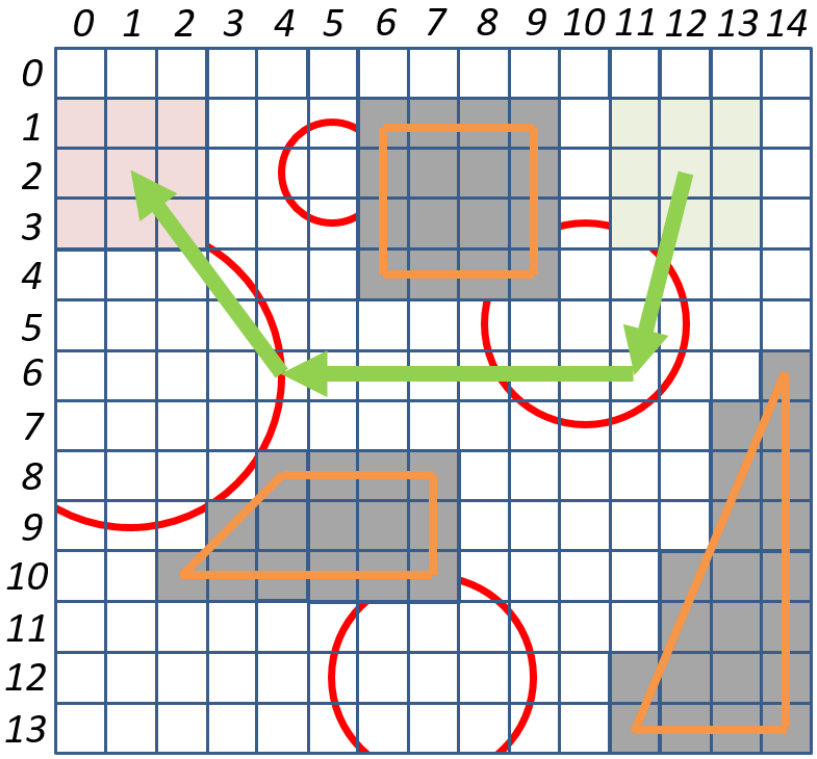
Окружность с центром в координатах (2,2) и радиусом 2.

Также на поле размещено M препятствий, проекции на поле которых можно представить в виде замкнутых ломаных кривых. Никакая часть проекции устройства на поле не может пересекаться с ячейкой занятой проекцией препятствия.

Для робототехнического устройства заданы координаты ячейки старта и ячейки, где устройство должно финишировать. Координаты ячейки старта и ячейки финиша совпадают с центром устройства.

Известно, что устройство - голономное, в любой момент времени перемещения оно занимает квадрат 3 на 3 ячейки, стороны квадрата всегда параллельны осям X и Y.

Необходимо найти кратчайший путь перемещения робототехнического устройства от места старта к месту финиша.



Если перемещение из координаты старта в координату финиша для устройства невозможно, то вывести −1.

Формат входных данных:

Первая строка содержит размер поля $K \times L$ $(5 \leq K, L \leq 32000)$. K - количество ячеек по горизонтали, L - количество ячеек по вертикали. Отсчет начинается с ячейки с координатами (0, 0).

Вторая строка содержит координаты S_1, S_2 места старта робототехнического устройства $(1 \leq S_1 \leq K-2), (1 \leq S_2 \leq L-2)$.

Третья строка содержит координаты F_1, F_2 места финиша робототехнического устройства $(1 \leq F_1 \leq K-2), (1 \leq F_2 \leq L-2)$.

Четвертая строка содержит два целых числа N и M - количество окружностей и количество препятствий $(3 \leq N, M \leq 100)$.

Следующие N строк описывают окружности. Каждая строчка содержит четыре целых числа H, C_1, C_2, R - координаты окружности и радиус $(1 \leq R \leq 100), (-99 \leq C_1, C_2 \leq 32099), (0 \leq H \leq N-1)$.

Дальше следует M блоков, состоящих из нескольких строк. Первая строка блока содержит два целых числа O и P - номер препятствия и количество вершин $(0 \leq O \leq M-1), (3 \leq P \leq 1000)$. Последующие P строк блока содержат по два целых числа U_1, U_2 - координаты вершин препятствия под номером O $(0 \leq U_1 \leq K-1), (0 \leq U_2 \leq L-1)$. Вершины одной ломаной перечисляются последовательно.

Формат выходных данных:

Выведите номера всех окружностей, разделенных пробелом, по которым проходить кратчайший путь перемещения из координаты начала в координату финиша. Считается, что путь робототехнического устройства, проходит через окружность, если какая-то часть устройства пересекается с одной из ячеек, принадлежащей окружности. Номера окружностей должны идти в том же порядке, в каком они будут посещены робототехническим устройством.

Пример:

stdin:

15 14
12 2
1 2
4 3
0 5 2 1
1 10 5 2
2 1 6 3
3 7 12 2
0 4
9 1
9 4
6 4
6 1
1 3
11 13
14 13
14 6
2 4
4 8
2 10
7 10
7 8

stdout:

1 2

Решение:

Решение состоит из 3 этапов.

На первом этапе необходимо аппроксимировать препятствия и окружности в целочисленные координаты, например, используя алгоритмы растровой дискретизации Брезенхема.

На втором этапе необходимо преобразовать поле в граф со следующими свойствами: вершины графа – те точки куда может попасть центр робота; у вершин графа указан номер окружности, которую робот будет касаться(пересекать) если окажется в данной точке; ребра графа имеют одинаковый вес.

На третьем этапе необходимо воспользоваться поиском пути по графу (например, поиск в ширину) и найти кратчайший путь, при этом определяя пересекаемые окружности для каждой вершины графа в процессе определения оптимального пути. В случае если имеется несколько путей, то следует выбирать тот, где окружностей меньше, т.е. возможно переопределение пути до любой точки в процессе поиска пути в графе.

Пример программы, реализующей данный алгоритм на языке Java:

```
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {

  private static boolean notObstacle(String s) {
    if (s.length() > 0)
      return !s.contains(".");
    else
      return false;
```

```
}

public static void main(String[] args)
    throws FileNotFoundException {
  double accuracy = 0.1;

  Scanner sc = new Scanner(System.in);
  int x = sc.nextInt();
  int y = sc.nextInt();
  String[][] field = new String[y][x];

  int[][] graph = new int[y][x];
  int[][][] graphA = new int[y][x][4];
  String[][] way = new String[y][x];
  for (int i = 0; i < y; i++) {// y
    for (int j = 0; j < x; j++) {// x
      way[i][j] = "";
      graph[i][j] = 0;
      graphA[i][j][0] = 0;// f(x)
      graphA[i][j][1] = 0;// g(x)
      graphA[i][j][2] = 0;// h(x)
      graphA[i][j][3] = 0;// nothing(0) / open (1) / close(2)

    }
  }

  for (int i = 0; i < y; i++) {
    for (int j = 0; j < x; j++) {
      field[i][j] = "--";
    }
  }
  int xBegin = sc.nextInt();
  int yBegin = sc.nextInt();
  field[yBegin][xBegin] = "bb";
  graph[yBegin][xBegin] = 2;

  int xEnd = sc.nextInt();
  int yEnd = sc.nextInt();
  field[yEnd][xEnd] = "ee";
  graph[yEnd][xEnd] = 3;

  int numOfCircle = sc.nextInt();
  int numOfObstacle = sc.nextInt();
  for (int i = 0; i < numOfCircle; i++) {
    int ni = sc.nextInt();
    int xi = sc.nextInt();
    int yi = sc.nextInt();
    int ri = sc.nextInt();
    for (int j = 0; j < y; j++) { // y
      for (int k = 0; k < x; k++) { // x
        if (((k - xi) * (k - xi)
            + (j - yi) * (j - yi)) <= (ri * ri))
          field[j][k] = " " + Integer.toString(ni);
      }
    }

  }
```

```
for (int i = 0; i < numOfObstacle; i++) { // reading obstacles
  int ni = sc.nextInt();
  int hmi = sc.nextInt();
  int xiOld = sc.nextInt();
  int yiOld = sc.nextInt();
  int x0 = xiOld;
  int y0 = yiOld;
  field[yiOld][xiOld] = "." + Integer.toString(ni);
  for (int j = 1; j < hmi; j++) {
    int xi = sc.nextInt();
    int yi = sc.nextInt();
    field[yi][xi] = "." + Integer.toString(ni);

    for (int k = 0; k < y; k++) { // y
      for (int l = 0; l < x; l++) { // x
        double a1 = Math.sqrt(
            Math.pow(l - xi, 2) + Math.pow(k - yi, 2));
        double a2 = Math.sqrt(Math.pow(l - xiOld, 2)
            + Math.pow(k - yiOld, 2));
        double a = Math.sqrt(Math.pow(xi - xiOld, 2)
            + Math.pow(yi - yiOld, 2));
        if (((a1 + a2 - a) < accuracy)
            && (l >= Math.min(xi, xiOld))
            && (l <= Math.max(xi, xiOld))
            && (k >= Math.min(yi, yiOld))
            && (k <= Math.max(yi, yiOld)))
          field[k][l] = "." + Integer.toString(ni);
      }
    }
    xiOld = xi;
    yiOld = yi;
  }

  for (int k = 0; k < y; k++) { // y
    for (int l = 0; l < x; l++) { // x
      double a1 = Math.sqrt(
          Math.pow(l - x0, 2) + Math.pow(k - y0, 2));
      double a2 = Math.sqrt(Math.pow(l - xiOld, 2)
          + Math.pow(k - yiOld, 2));
      double a = Math.sqrt(Math.pow(x0 - xiOld, 2)
          + Math.pow(y0 - yiOld, 2));
      if (((a1 + a2 - a) < accuracy)
          && (l >= Math.min(x0, xiOld))
          && (l <= Math.max(x0, xiOld))
          && (k >= Math.min(y0, yiOld))
          && (k <= Math.max(y0, yiOld)))
        field[k][l] = "." + Integer.toString(ni);
    }
  }
}

for (int i = 1; i < (y - 1); i++) {// y
  for (int j = 1; j < (x - 1); j++) {// x
    if ((graph[i][j] == 0)
        && notObstacle(field[i - 1][j - 1])
        && notObstacle(field[i - 1][j])
```

```
          && notObstacle(field[i - 1][j + 1])
          && notObstacle(field[i][j - 1])
          && notObstacle(field[i][j])
          && notObstacle(field[i][j + 1])
          && notObstacle(field[i + 1][j - 1])
          && notObstacle(field[i + 1][j])
          && notObstacle(field[i + 1][j + 1]))
        graph[i][j] = 1;
    }
  }

  for (int i = 0; i < y; i++) {// y
    for (int j = 0; j < x; j++) {// x
      if (graph[i][j] == 0)
        graphA[i][j][3] = 2;
    }
  }

  int xi = xBegin;
  int yi = yBegin;
  int minFx = 300000000;
  int xiMin = xBegin;
  int yiMin = yBegin;
  graphA[yi][xi][1] = 0;
  graphA[yi][xi][2] = 10
      * (Math.abs(xEnd - xi) + Math.abs(yEnd - yi));
  graphA[yi][xi][0] = graphA[yi][xi][2];
  graphA[yi][xi][3] = 2;

  String circles = "";
  String s;
  int count = 0;

  int xiOld = -1;
  int yiOld = -1;
  boolean found = true;
  while (((xi != xEnd) || (yi != yEnd)) && found) {
    graphA[yi][xi][3] = 2;

    if ((xiOld == xi) && (yiOld == yi)) {
      found = false;
      for (int i = 1; i < y - 1; i++) {
        for (int j = 1; j < x - 1; j++) {
          if (!found && graphA[i][j][3] == 1) {
            xi = j;
            yi = i;
            found = true;
            circles = way[i][j];
            graphA[yi][xi][3] = 2;
            break;
          }
        }
      }
    }
    minFx = 300000000;

    for (int i = 0; i < 3; i++) {
```

```
  for (int j = 0; j < 3; j++) {
    if (graphA[yi + i - 1][xi + j - 1][3] != 2) {
      if (graphA[yi + i - 1][xi + j - 1][3] == 0) {
        way[yi + i - 1][xi + j - 1] = circles;
        graphA[yi + i - 1][xi + j - 1][3] = 1;

        if ((j == 1) || (i == 1))
          graphA[yi + i - 1][xi + j
              - 1][1] = graphA[yi][xi][1] + 10;
        else
          graphA[yi + i - 1][xi + j
              - 1][1] = graphA[yi][xi][1] + 14;
        graphA[yi + i - 1][xi + j - 1][2] = Math
            .toIntExact(Math.round(10 * Math.sqrt(Math
                .pow(xEnd - (xi + j - 1), 2)
                + Math.pow(yEnd - (yi + i - 1), 2))));
        graphA[yi + i - 1][xi + j
            - 1][0] = graphA[yi + i - 1][xi + j
                - 1][1]
            + graphA[yi + i - 1][xi + j - 1][2];
      } else { // graphA[yi + i - 1][xi + j - 1][3] == 1
        if ((j == 1) || (i == 1)) {
          if ((graphA[yi][xi][1]
              + 10) < graphA[yi + i - 1][xi + j
                  - 1][1]) {
            graphA[yi + i - 1][xi + j
                - 1][1] = graphA[yi][xi][1] + 10;
            graphA[yi + i - 1][xi + j
                - 1][0] = graphA[yi + i - 1][xi + j
                    - 1][1]
                    + graphA[yi + i - 1][xi + j
                        - 1][2];

            way[yi + i - 1][xi + j - 1] = circles;
          }
        } else {
          if ((graphA[yi][xi][1]
              + 14) < graphA[yi + i - 1][xi + j
                  - 1][1]) {
            graphA[yi + i - 1][xi + j
                - 1][1] = graphA[yi][xi][1] + 14;
            graphA[yi + i - 1][xi + j
                - 1][0] = graphA[yi + i - 1][xi + j
                    - 1][1]
                    + graphA[yi + i - 1][xi + j
                        - 1][2];

            way[yi + i - 1][xi + j - 1] = circles;
          }
        }
      }
    }
  }
}

for (int i = 1; i < y - 1; i++) {
  for (int j = 1; j < x - 1; j++) {
```

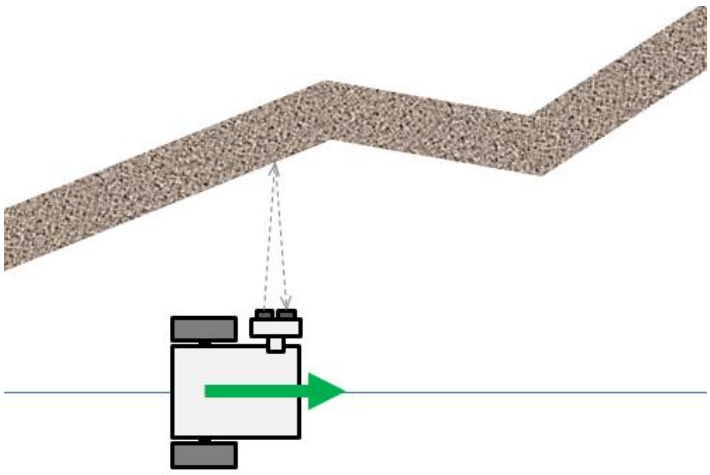
```
            if ((graphA[i][j][3] == 1)
                && (minFx > graphA[i][j][0])) {
              circles = way[i][j];
              minFx = graphA[i][j][0];
              xiMin = j;
              yiMin = i;
            }
          }
        }

        xiOld = xi;
        yiOld = yi;

        xi = xiMin;
        yi = yiMin;

        for (int i = 0; i < 3; i++) {
          for (int j = 0; j < 3; j++) {
            s = field[yi + i - 1][xi + j - 1];
            if (s.contains(" ") && !circles.contains(s))
              circles += s;
          }
        }

      }

      if (!((xi == xEnd) && (yi == yEnd)))
        circles = "";

      if (!circles.isEmpty())
        System.out.println(circles);
      else
        System.out.println("-1");

    }
  }
```

Задача 2.3.5 (15 баллов)

Условие:

Робототехническое устройство, собранное по дифференциальной схеме и с заданным диаметром колес, движется по прямой с постоянной скоростью. На устройстве установлен ультразвуковой датчик расстояния, направленный влево перпендикулярно направлению движения.

Слева от робототехнического устройства располагается стена, состоящая из нескольких сегментов S $(1 \leq S \leq 5)$. Каждый сегмент имеет свою протяженность и расположен под углом относительно предыдущего сегмента. Минимальная длина сегмента - 380 мм, максимальная длина сегмента 760 мм, угол между плоскостью сегмента и плоскостью, проходящей перпендикулярно через ось траекторию движения робота, не больше 45 градусов. Между сегментами нет зазоров. Во время движения датчик расстояния возвращает расстояние до сегмента стены, мимо которого проезжает робот. Известно, что показания ультразвукового датчика не идеальны, в них всегда присутствует высокая доля помех. Частота опроса датчика - 10 Гц.



Нужно найти количество сегментов в стене S.

Формат входных данных:

Первая строка входных данных содержит два целых числа d, n - диаметр колес d $(20 \leq d \leq 100)$ в миллиметрах и количество показаний ультразвукового датчика расстояния $(10 \leq n \leq 10^3)$. В следующих n строках перечисляются пары чисел E, L - среднее арифметическое показаний энкодеров левого и правого колеса в градусах $(0 \leq E \leq 2^{31} - 1)$ и показание датчика расстояния в миллиметрах $(30 \leq L \leq 2550)$. Угол поворота колеса соответствует углу поворота энкодера.

Формат выходных данных:

Выведите единственное число - количество сегментов в стене.

Пример:

stdin:

28 325
7 198
17 197
29 190
40 197
52 205
63 198
76 197
88 197
100 190
112 190
123 190
136 199
147 195
159 190
171 190
183 195
. . .
3626 184
3638 211
3650 191
3661 189
3673 184
3685 191

3696 195
3708 190
3719 179
3729 183
3740 184
3750 184
stdout:
3

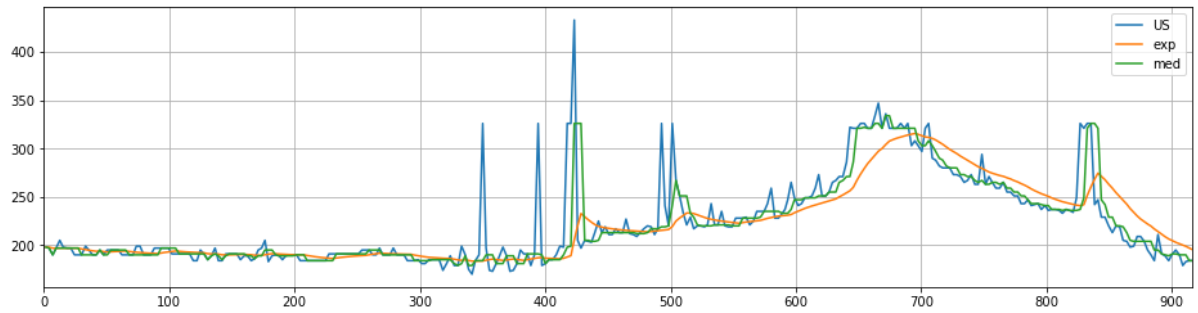
Решение:

Задача относится к классу задач на анализ данных.

Чтобы восстановить количество сегментов, из которых составлена стена, определить в последовательности измерений границы участков, где происходит изменение прироста в значениях датчика. При этом возможны следующие варианты:

- соседние показания датчика были примерно одинаковые, потом начали линейно возрастать;
- соседние показания датчика линейно возрастали, потом начали линейно убывать;
- соседние показание датчика линейно убывали, потом стали примерно равны друг другу;
- и т.п.

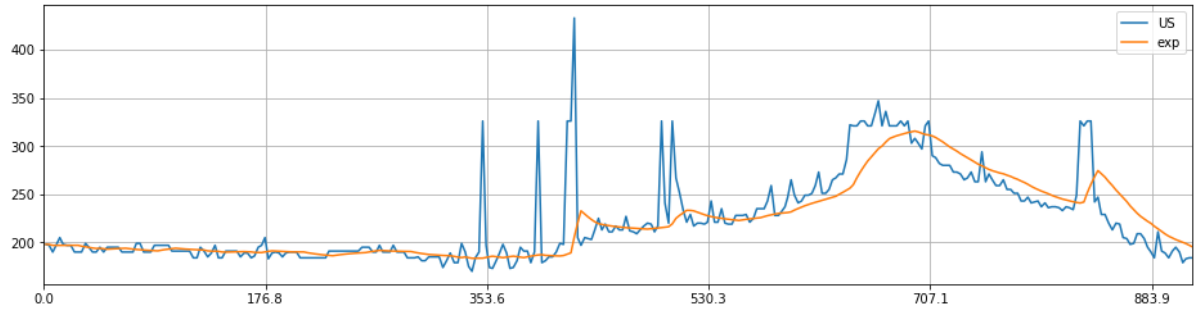
Поскольку данные зашумлены, мы не можем сравнивать соседние показания датчиков друг с другом. Поэтому необходимо использовать фильтрацию для уменьшения шума. Параметры фильтрации нужно выбирать так, чтобы, с одной стороны, чрезмерные всплески исчезли, но, с другой стороны, переходы между сегментами однозначно идентифицировались.



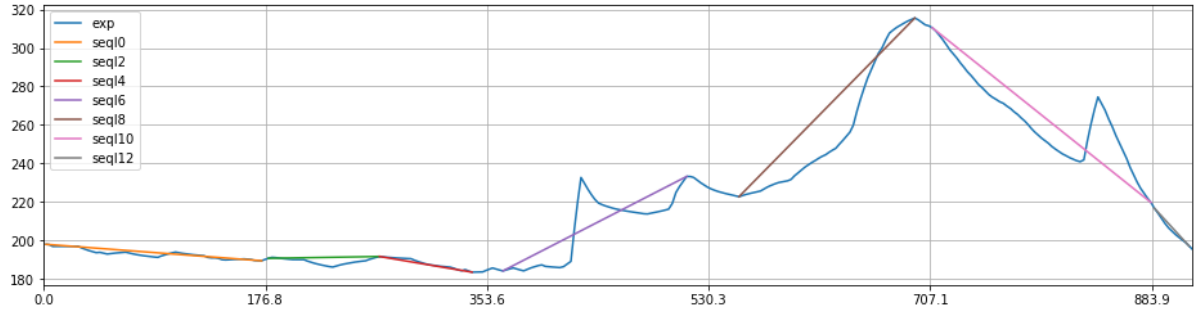
Для получения графиков, представленных выше использовались следующие фильтры:

- зеленая кривая - медианный фильтр с окном в 5 измерений;
- оранжевая кривая - последовательное применение медианного фильтра с окном в 5 измерений и экспоненциальное скользящее среднее с коэффициентом стабилизации 0.88.

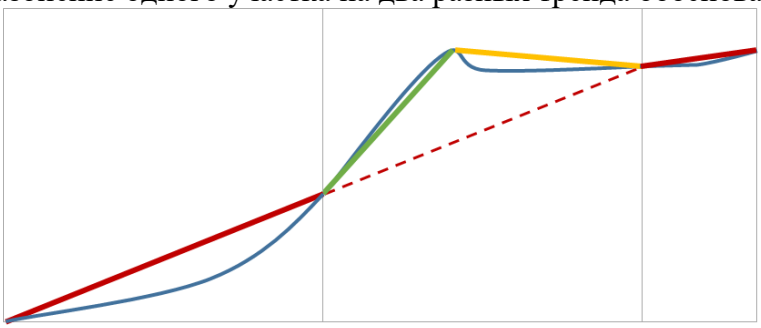
После фильтрации полученные данные все равно не готовы к сравнению между собой - на небольшом участке данных они все еще могут изменять направление, хотя принадлежат одному сегменту. Поэтому нужно сделать укрупнение исследуемых участков: сравнивать между собой не соседние показания, а тренды поведения графика на соседних участках. Для этого нужно сделать разбиение всех показаний на группы. Размер группы разумно сделать соизмеримым с величиной сегмента. Но поскольку длина сегмента строго не зафиксирована, согласно условию задачи, то нужно опираться на его минимальное возможное значение. В ходе решения задачи выяснилось, что удобно оперировать сегментами длиной в 250 мм. Тогда проекция сегмента на ось X, определяющая количество измерений принадлежащих этому сегменту, будет равна $l_{segment} \cdot \cos\frac{\pi}{4}$.



Теперь, если на каждом участке взять крайнее левое и крайнее правое значения, то можно определить тренд прироста показаний. При этом, такое построение тренда в случаях значительного искривления графика будет приводить к неправильному определению частного направления изменения показаний. Следовательно, нужно использовать комплексный подход, основанный на получении экстремумов - максимума и минимума показаний на каждом участке.



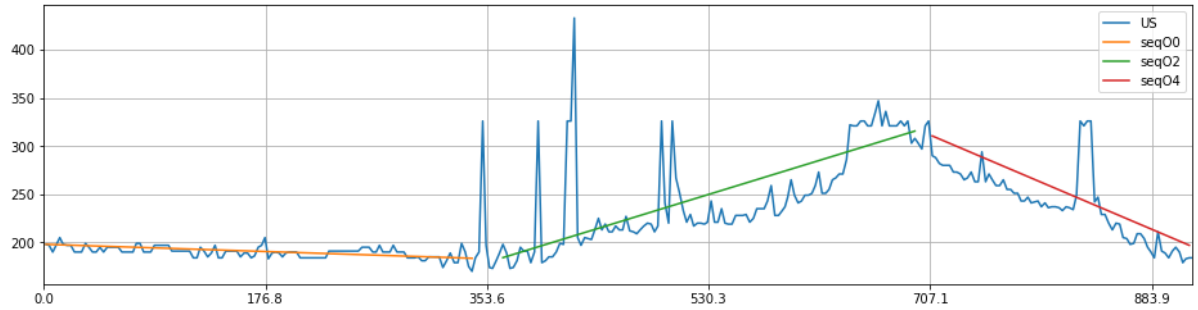
Вариант, когда разбиение одного участка на два разных тренда обосновано:



Появление линий трендов позволяет сравнивать участки между собой. Для этого для каждой линии определяется ее угловой коэффициент:

$$k = \frac{y_2 - y_1}{x_2 - x_1}$$

- Если коэффициенты соседних участков равны по знаку, то можно эти участки объединить, предполагая, что они принадлежат одному и тому же сегменту, а разница в значении коэффициентов вызвана сглаживанием.
- Если коэффициенты соседних участков отличаются по знаку, то они принадлежать двум разным сегментам. Исключение можно делать для участков, угол между линий трендов которых меньше 10 градусов, поскольку такое отклонение может быть также вызвано сглаживанием.



Количество участков, полученных после объединения, определяет количество сегментов в стене, рядом с которой ехал робот.

Пример программы, реализующей данный алгоритм на языке Python:

```
import math
import numpy as np

MEDIAN_WINDOW_SIZE=5
K_ema = 0.88
SLOPE_MIN_THRESHOLD=10
SLOPE_MAX_THRESHOLD=45-SLOPE_MIN_THRESHOLD

#Calculate how many degrees corresponds the slope of a segment
def calc_deg_of_slope(pos1, pos2):
    return math.atan2(pos2[1]-pos1[1],
                      pos2[0]-pos1[0])*180/math.pi

#minimal length of the segment
seg_length=250.0
seg_proj=seg_length*math.cos(math.pi/4)

(diameter, num_of_measurements) = input().split()

#Read measurements and filter them
window = []
values = []
for i in range(0, int(num_of_measurements)):
    (enc, value) = input().split()
    #Use millimeters instead of encoder ticks
    x = float(enc)*math.pi*int(diameter)/360
    us = float(value)
    window.append(us)
    if i >= MEDIAN_WINDOW_SIZE-1:
        #median filter implementation
        win_sorted = window[:]
        win_sorted.sort()
        us = win_sorted[2]
        del window[0]
    if i != 0:
        #exponential moving average filter implementation
        us = values[i-1][1] * K_ema + us * (1 - K_ema)
    values.append([x, us])
    i += 1

#Identification of all meaningful segments
pos_list = []
start_pos = 0
pos_b = values[0][:]
```

```
pos_e = []
pos_min = pos_b[:]
pos_max = pos_b[:]
for row in values:
    #initial segmentation is based on minimal segment size:
    #looking for min and max measurements to make
    #an assumption about the slope of the segment
    if (row[0] < start_pos + seg_proj) and (row != values[-1]):
        pos_e = row[:]
        if row[1] > pos_max[1]:
            pos_max = row[:]
        if row[1] < pos_min[1]:
            pos_min = row[:]
    else:
        if pos_max[0] < pos_min[0]:
            pos1 = pos_max[:]
            pos2 = pos_min[:]
        else:
            pos1 = pos_min[:]
            pos2 = pos_max[:]
        #if the distance between min and max is small
        #it is not reliable since it could be impacted by noise
        #so the parts belonging to the segment but outside
        #the min-max range could be used if they are not short
        if (pos2[0] - pos1[0] < seg_proj/2):
            seg_left=pos1[0] - pos_b[0]
            seg_right=pos_e[0] - pos2[0]
            if (pos1[0] - pos_b[0]) > (pos_e[0] - pos2[0]):
                if seg_left > seg_proj/2:
                    pos_list.append(pos_b)
                    pos_list.append(pos1)
                pos_list.append(pos1)
                pos_list.append(pos2)
            else:
                pos_list.append(pos1)
                pos_list.append(pos2)
                if seg_right > seg_proj/2:
                    pos_list.append(pos2)
                    pos_list.append(pos_e)
        else:
            pos_list.append(pos1)
            pos_list.append(pos2)

        #ready to consider the next segment
        start_pos += seg_proj
        pos_b = row[:]
        pos_e = []
        pos_min = pos_b[:]
        pos_max = pos_b[:]

#investigate slopes of the segments:
#1. consider two segements as one if slopes of both
#   segments are in the same direction
#   except the cases when one segment is steeply
#   sloping and another is gently sloping
#2. consider two segments as one if slopes are
#   in different direction but both are gentle
```

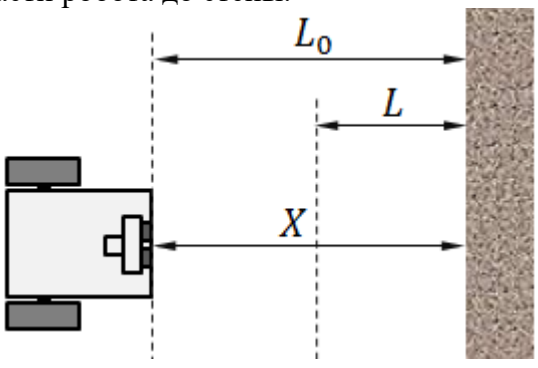
```
new_list = [pos_list[0]]
last_elem = 1
for i in range(1,len(pos_list)-1,2):
    k_left  = calc_deg_of_slope(new_list[-1], pos_list[i])
    k_right = calc_deg_of_slope(pos_list[i+1], pos_list[i+2])
    if np.sign(k_left) != np.sign(k_right):
        if (np.fabs(k_left-k_right)>SLOPE_MIN_THRESHOLD):
            new_list.append(pos_list[i])
            new_list.append(pos_list[i+1])
    elif (np.fabs(k_left-k_right)>SLOPE_MAX_THRESHOLD):
        new_list.append(pos_list[i])
        new_list.append(pos_list[i+1])
    last_elem=i+2
new_list.append(pos_list[last_elem])

#calculate number of assumed segments excluding short
#segments
c = 0
for i in range(0,len(new_list),2):
    if new_list[i+1][0] - new_list[i][0] >= seg_proj/2:
        c += 1
print(c)
```

Задача 2.3.6 (15 баллов)

Условие:

Перед стеной по направлению к стене установлено робототехническое устройство массой 1 кг. Устройство спроектировано так, что оно приводится в движение только одним мотором, который вращает два колеса диаметром D мм., расположенных слева и справа от устройства. Колеса вращаются синхронно. Угол поворота вала мотора совпадает с углом поворота колеса. Таким образом, устройство может либо приближаться, либо удаляться от стены. При движении на устройство действует сила сопротивления среды (сопротивление воздуха), которая пропорциональна текущей скорости перемещения. Коэффициент пропорциональности силы сопротивления среды $K_{friction}$.

На передней части устройства расположен датчик дальномер. Изначально устройство установлено у стены на таком расстоянии, что датчик показывает L_0 мм. После начала движения информация с датчика считывается каждые 50 миллисекунд. Датчик работает идеально точно - в любой момент времени показывает кратчайшее расстояние от передней части робота до стены.



Перед устройством ставится задача - переместиться на расстояние L мм. от стены. Устройство управления роботом запрограммировано таким образом, что оно генерирует значение крутящего момента M на моторе согласно закону

где k - коэффициент пропорциональности при управлении моментом силы мотора, а X - Текущее показание датчика дальномера. Момент силы меняется дискретно, одновременно с измерением расстояния датчиком.

Необходимо найти значение коэффициента k с точностью 0.1, при котором робототехническое устройство остановится на заданное расстояние за минимальное время. Критерием остановки считать условие $|L - X| + |v| < 10$, где X- текущее показание датчика дальномера, а v - Текущая скорость устройства в миллиметрах в секунду.

При решении задачи необходимо учитывать, что устройство разрушится при столкновении со стеной и не сможет продолжать движение.

Подготовьте решение задачи в общем виде и загрузите файл с входными данными, чтобы найти конкретное значение k и соответствующее ему минимальное время остановки t, Для подготовки решения задачи в общем виде можете использовать информацию о крайних значениях параметров и шаг их изменения:

- $0.1 \leq K_{friction} \leq 4$ с шагом 0.1, Коэффициент используется для скорости, выраженной в м/с.;
- $20 \leq D \leq 100$ (в миллиметрах) с шагом 5;
- $100 \leq L_0 \leq 600$ (в миллиметрах) с шагом 50;
- $L = L_0 + a$, где $-400 \leq a \leq 100$ (в миллиметрах) с шагом 50, при этом L не будет меньше 50

Формат входных данных:

Файл содержит строчки:

- Первая строчка содержит значение $K_{friction}$
- Вторая строчка содержит значение D
- Третья строчка - значение L_0
- Четвертая строчка - значение L

Формат выходных данных:

Введите два числа в одну строку:

- первое число содержит значение коэффициента k с точностью до десятых;
- второе число содержит время остановки t с точностью до сотых.

Числа разделены одним пробелом, целая часть от дробной отделена точкой.

Решение:

Для решения задачи нужно построить математическую модель описанной системы.

Из условия задачи известно, что робот движется только в одном направлении, а, следовательно, движение - прямолинейное, так же, известно, что на систему действует коэффициент пропорциональности силы сопротивления среды, поэтому можно сказать, что движение - равноускоренное.

Из общей формулы равноускоренного-прямолинейного движения можно определить положение робота в каждый момент времени, а значит и крутящий момент, который будет задаваться мотором.

$$x = x_0 + v_{0x}t + \frac{a_x t^2}{2}$$

Поскольку и скорость и ускорение будут меняться со временем, то удобнее преобразовать формулу к

$$x_n = x_{n-1} + v_{xn-1}\Delta t + \frac{a_x \Delta t^2}{2}$$

где Δt - фиксировано и равно 0,05, x_0 - равно L_0, v_{x0} равно 0. При этом в каждый момент времени:

$$v_n = v_{xn-1} + a_x \Delta t$$

$$a = \frac{F}{m}$$

$$F = F_{motor} - F_{friction}, F_{motor} = \frac{M}{r}, r = \frac{D}{2}, F_{friction} = K_{friction} \cdot v$$

$$F = \frac{2M}{D} - K_{friction} v$$

В формуле выше момент силы M вычисляется в зависимости от расстояния робота до препятствия x_{n-1} и зависит от неизвестного коэффициента k.

Чтобы найти ответ на задачу нужно перебирать все возможные значения коэффициента k, затем моделировать процесс изменения X в зависимости от изменяющихся значений момента силы. При этом нужно отслеживать, что значение X не стало меньше 0, что означает столкновение с препятствием. Если устройство в какой-то момент времени оказывается близко к требуемой позиции (L) и скорость его перемещения мала, то данный коэффициент и время, необходимое для достижения условия остановки запоминается.

После окончания перебора всех возможных значений k, нужно из запомненных значений выбрать то, которое приводило к остановке быстрее всего.

Пример программы, реализующей данный алгоритм на языке Python:

```
def regulation(l,x,k):
    motor = (l-x)*k
    if motor >= 10 :
        motor = 10
    elif motor <= -10 :
        motor = -10
    return motor

def findK(dataset):
    ll =dataset.split()
    l = int(ll.pop())/ 1000
    l0 = int(ll.pop())/ 1000
    d = int(ll.pop())/ 1000
    kf = float(ll.pop())
    k = 0
    t = 0.05
    kmin = 200
    tmin = -1
    flag = False
    while k <= 200:
        k += 0.1
        v = 0
        x = l0
        tm = 0
        v0 = 0
        count = 0
        while (abs(l-x) + abs(v))*1000 > 10:
            M = regulation(l, x, k)
            v0 = v
            a = 2 * M / d - kf * v
            v = v0 + a * t
            x = x + v0 * t + a * t * t / 2
            tm += t
```

```
            count += 1
            if x < 0 or count > 1000000:
                flag = True
                break
        if not flag and ( tmin < 0 or tmin > tm) :
            tmin = tm
            kmin = k
    return '%.1f %.2f' % (kmin, tmin)

print(findK(sys.stdin.read()))
```